# Link prediction in bipartite multi-layer networks, with an application to drug-target interaction prediction

Maksim Koptelov

Normandie Université

# THÈSE

**Pour obtenir le diplôme de doctorat**

**Spécialité INFORMATIQUE**

**Préparée au sein de l'Université de Caen Normandie**

# Link prediction in bipartite multi-layer networks, with an application to drug-target interaction prediction


**Présentée et soutenue par**
**Maksim KOPTELOV**


| Thèse soutenue publiquement le 30/09/2020 devant le jury composé de | | |
|---|---|---|
| M. TIJL DE BIE | Professeur des universités, Université de Ghent | Rapporteur du jury |
| Mme CHRISTEL VRAIN | Professeur des universités, Université d'Orléans | Rapporteur du jury |
| M. ERWIN BAKKER | Senior researcher, Leiden University | Membre du jury |
| Mme CELINE ROUVEIROL | Professeur des universités, Université Paris 13 Paris-Nord | Président du jury |
| M. ALBRECHT ZIMMERMANN | Maître de conférences, Université Caen Normandie | Membre du jury |
| M. BRUNO CREMILLEUX | Professeur des universités, Université Caen Normandie | Directeur de thèse |
| Mme LINA FATIMA SOUALMIA | Maître de conférences HDR, Université Rouen Normandie | Co-directeur de thèse |

**Thèse dirigée par BRUNO CREMILLEUX et LINA FATIMA SOUALMIA, Groupe de recherche en informatique, image, automatique et instrumentation**

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC

Dedicated to my mother.

*Second printing, October 2020*

## Abstract

Many aspects from real life with bi-relational structure can be modeled as bipartite networks. This modeling allows the use of some standard solutions for prediction and/or recommendation of new relations between these objects in such networks. Known as the link prediction task, it is a widely studied problem in network science for single graphs, networks assuming one type of interaction between vertices. For multi-layer networks, allowing more than one type of edges between vertices, the problem is not yet fully solved.

The motivation of this thesis comes from the importance of an application task, drug-target interaction prediction. Searching valid drug candidates for a given biological target is an essential part of modern drug development. In this thesis, the problem is modeled as link prediction in a bipartite multi-layer network. Modeling the problem in this setting helps to aggregate different sources of information into one single structure and as a result to improve the quality of link prediction.

The thesis mostly focuses on the problem of link prediction in bipartite multi-layer networks and makes two main contributions on this topic. The first contribution provides a solution for solving link prediction in the given setting without limiting the number and type of networks, the main constrains of the state of the art methods. Modeling random walk in the fashion of PageRank, the algorithm that we developed is able to predict new interactions in the network constructed from different sources of information. The second contribution, which solves link prediction using community information, is less straight-forward and more dependent on fixing the parameters, but provides better results. Adopting existing community measures for link prediction to the case of bipartite multi-layer networks and proposing alternative ways for exploiting communities, the method offers better performance and efficiency. Additional evaluation on the data of a different origin than drug-target interactions demonstrate the genericness of proposed approach.

In addition to the developed approaches, we propose a framework for validation of predicted interactions founded on an external resource. Based on a collection of biomedical concepts used as a knowledge source, the framework is able to perform validation of drug-target pairs using proposed confidence scores. An evaluation of predicted interactions performed on unseen data shows effectiveness of this framework.

At the end, a problem of identification and characterization of promiscuous compounds existing in the drug development process is discussed. The problem is solved as a machine learning classification task. The contribution includes graph mining and sampling approaches. In addition, a graphical interface was developed to provide feedback of the result for experts.

## Résumé

De nombreux problèmes réels relèvent d'une structure bi-relationnelle et peuvent être modélisés suivant des réseaux bipartis. Une telle modélisation permet l'utilisation de solutions standards pour la prédiction et/ou la recommandation de nouvelles relations entre objets de ces réseaux. La tâche de prédiction de liens est un problème largement étudié dans les réseaux simples, c'est-à-dire les réseaux avec un seul type d'interaction entre sommets. Cependant, pour les réseaux multicouche (i.e. réseaux avec plusieurs types d'arêtes entre sommets), ce problème n'est pas encore entièrement résolu.

Cette thèse est motivée par l'importance d'une tâche réelle, à savoir la prédiction d'interaction entre un médicament et une cible thérapeutique. La recherche de candidats médicaments prometteurs pour une cible thérapeutique biologique donnée est une partie essentielle de la conception d'un médicament moderne. Dans cette thèse, nous modélisons ce problème comme une tâche de prédiction de lien dans un réseau multicouche biparti. Cette modélisation du problème permet de rassembler différentes sources d'information en une seule structure et ainsi d'améliorer la qualité de la prédiction d'un lien.

Cette thèse se concentre sur le problème de la prédiction de liens dans les réseaux multicouches bipartis et apporte deux contributions principales à ce sujet. La première contribution est une solution pour résoudre la prédiction de liens sans limiter le nombre et le type de réseaux, ce qui est le principal défaut des méthodes de l'état de l'art. L'algorithme que nous avons développé modélise une marche aléatoire à la manière du PageRank et est capable de prédire de nouvelles interactions dans le réseau que nous construisons à partir de différentes sources d'information. La deuxième contribution, qui porte aussi sur ce problème, s'appuie sur les méthodes de détection de communautés. Cette solution, moins immédiate et plus dépendante du choix des valeurs des paramètres, donne de meilleurs résultats. Pour cela, nous adaptons des mesures utilisées pour la détection de communautés à la problématique de la prédiction de liens dans les réseaux multicouche bipartis et nous développons de nouvelles méthodes associant des communautés pour la prédiction de liens. Nous évaluons aussi nos méthodes sur des données autres que celles des interactions entre médicaments et cibles thérapeutiques montrant ainsi le caractère générique de notre approche.

D'autre part, nous proposons un protocole expérimental de validation des interactions prédites reposant sur l'exploitation de ressources externes. Fondé sur une collection de concepts biomédicaux utilisés comme source de connaissances, ce protocole effectue une validation des paires de médicaments-cibles thérapeutiques qui sont prédites à partir de scores de confiance que nous avons définis. Une évaluation des interactions prédites sur des données tests montre l'efficacité de ce protocole.

Enfin, nous nous intéressons au problème de l'identification et de la caractérisation de composés promiscues qui existe dans le processus de développement de médicaments. Nous modélisons ce problème comme une tâche de classification et le résolvons par l'apprentissage automatique. Notre contribution repose sur une approche d'exploration de graphes et d'échantillonnage. De plus, nous avons développé une interface graphique pour fournir un retour d'information aux experts sur les résultats.

# Contents


**Introduction** . . . . . 1

**Context** 1

Link prediction in bipartite multi-layer graphs . . . . . 1

Modern drug development . . . . . 2

**Contributions** 3

**Structure of the thesis** 4

**I Preliminaries**

**1 Foundations** . . . . . 7

**1.1 Definitions** 7

1.1.1 Chemoinformatic concepts . . . . . 7

1.1.2 Graph definitions . . . . . 9

1.1.3 Prediction quality metrics . . . . . 15

**1.2 Problem setting** 16

1.2.1 Survey on drug-target prediction . . . . . 17

1.2.2 Problem definition . . . . . 19

**2 Data sets** . . . . . 21

**2.1 Drug-target interactions** 21

2.1.1 The IUPHAR network . . . . . 21

2.1.2 Benchmark sets . . . . . 24

**2.2 Generic data sets** 25

2.2.1 MovieLens . . . . . 25

2.2.2 Unicode languages . . . . . 28

3 Validation of link prediction founded on external knowledge .... 31
3.1 Principle 31
3.1.1 The UMLS .... 31
3.1.2 The validation idea .... 34
3.1.3 Assessing the strength of interactions .... 34
3.2 The framework to validate link prediction 39
3.2.1 The algorithm .... 39
3.2.2 Result normalization .... 40
3.2.3 Validation example .... 41
3.3 Conclusion 42

II Predicting links with random walks

4 Classical random walker .... 45
4.1 Related work 45
4.1.1 State of the art .... 45
4.1.2 The random walk model for link prediction .... 46
4.2 Random walks on multi-layer network 47
4.2.1 Explicit Random Walker .... 47
4.2.2 Example .... 49
4.3 Conclusion 50

5 PageRank models .... 51
5.1 Related work 51
5.1.1 State of the art .... 51
5.1.2 PageRank algorithm .... 52
5.1.3 PageRank for link prediction .... 54
5.1.4 The NRWRH algorithm .... 55
5.2 The NEWERMINE algorithm 56
5.2.1 The algorithm .... 56
5.2.2 Computational example .... 58
5.3 Conclusion 61

6 Experimental evaluation of random walk methods .... 63
6.1 Experimental settings 63
6.1.1 Evaluation Protocol .... 63
6.1.2 Quality Measures .... 64
6.1.3 Implementation .... 64
6.2 Experimental results 64
6.2.1 Using three-layer graphs .... 64
6.2.2 Using the full graph with six layers .... 66
6.3 Sparsification experiment 67
6.3.1 Idea of the experiment .... 67
6.3.2 Results and discussion .... 68

**6.4 Validation of link prediction with external knowledge 69**
6.4.1 Experimental setup . . . 69
6.4.2 Experimental results . . . 69
**6.5 Conclusion 70**

**7 Analysis of connectivity characteristics . . . 71**
**7.1 Motivation 71**
**7.2 Experimental settings 72**
7.2.1 The data . . . 72
7.2.2 Experimental protocol . . . 72
**7.3 Experimental evaluation 73**
7.3.1 Link prediction in benchmark data sets . . . 73
7.3.2 Connectivity characteristics and redundancy . . . 74
7.3.3 Informativeness of networks . . . 77
7.3.4 Comparison with IUPHAR . . . 79
**7.4 Conclusion 79**

**III Predicting links with communities**

**8 Link prediction via community detection . . . 83**
**8.1 Related work 83**
8.1.1 State of the art . . . 83
8.1.2 Link prediction measures . . . 85
8.1.3 Community detection methods . . . 87
**8.2 The LPBYCD approach 89**
8.2.1 Link prediction in a bipartite setting . . . 90
8.2.2 Adapting prediction measures . . . 92
8.2.3 Auto-tuning of parameters . . . 96
8.2.4 Example . . . 96
**8.3 Conclusion 99**

**9 Experimental evaluation of predicting links with communities . . . 101**
**9.1 Experimental setup 101**
9.1.1 Parameter selection . . . 101
9.1.2 Experimental protocol . . . 102
**9.2 Experimental results 103**
9.2.1 Link prediction measures evaluation . . . 103
9.2.2 Parameter selection via internal cross-validation . . . 105
9.2.3 Comparison with a random walk method . . . 106
9.2.4 Comparison to the state of the art . . . 106
9.2.5 Performance on a bigger data set . . . 107
9.2.6 Running times and scalability . . . 108
9.2.7 Evaluation of the genericness . . . 109
9.2.8 Interpretability . . . 111

**9.3 Validation of link prediction with external knowledge 112**
9.3.1 Experimental settings . . . 112
9.3.2 Experimental results . . . 112
**9.4 Conclusion 114**

IV **Detecting promiscuous compounds**

10 **Identification and characterization of promiscuous compounds 117**
**10.1 Introduction 117**
10.1.1 Motivation . . . 117
10.1.2 Context/Definitions . . . 118
10.1.3 The idea . . . 119
**10.2 Related work 120**
10.2.1 PAINS identification/characterization . . . 120
10.2.2 (Q)SAR modeling . . . 120
10.2.3 Pattern mining . . . 120
**10.3 Substructure mining and model learning 121**
10.3.1 Discriminative subgraph mining . . . 121
10.3.2 Balancing the data . . . 121
10.3.3 The predictive model . . . 122
**10.4 Experimental evaluation 122**
10.4.1 Data preparation . . . 123
10.4.2 Experimental setup . . . 123
10.4.3 Performance evaluation for FH classification . . . 124
10.4.4 Model evaluation on randomly sampled compounds . . . 126
10.4.5 Model evaluation on DCM . . . 127
**10.5 Graphical interface 127**
**10.6 Conclusion 128**

**Conclusion and future directions . . . 131**

**Publications and prototypes . . . 135**

**Acknowledgments . . . 137**

**Bibliography . . . 139**

**Appendices . . . 151**

# Introduction

## Context

The title of this thesis points to two separate, but tightly connected problems emerging from each other: link prediction in a specific, but important, type of networks, bipartite and multi-layer at the same time, and its application for drug-target interaction prediction, a vital part of research in modern drug discovery. First of all, I would like to introduce the first objective, link prediction in bipartite multi-layer networks. I will explain the importance of studying graphs in general and for the given setting in particular, explain why I have chosen this setting, list its challenges and the solutions I develop in this thesis to address these challenges. Then I shortly describe the modern drug development process and explain how my application question is positioned in this process.

### Link prediction in bipartite multi-layer graphs

Many concepts from real life, for example inhabited places and their connections on road maps, people and their friendships in social networks, proteins and their interactions in protein interaction networks, can be represented by graphs, special mathematical structures to describe objects and their relations. The advantage of representing these objects as graphs is in the available tools that can be used to solve standard problems. In the above example, to propose new roads between cities, suggest new friendships for a circle of friends, or predict new interacting pairs of proteins (which has its own importance in research biology), existing techniques from network science, such as for link prediction, can be applied to solve it. Many of these applications can be modeled as a bipartite graph, vertices of which are divided into two distinct groups [152]. They are the tasks of user-product recommendation, member-club recommendation, authors-venues recommendation *etc.* In other words, many recommendation systems fall into this category [100]. The problem setting that motivates my work also falls in the same group – the prediction of links between drug candidates and biological targets, an essential step of computational drug development. This is why I model this problem as a link prediction problem in a bipartite graph.

A common challenge which the data on drug-target interaction prediction has is sparsity. It introduces difficulties or makes it impossible to use some common recommendation systems techniques for making predictions. As a proposed solution, I represent different sources of information

as separate networks (also called *layers* in this thesis), combine them into one single structure, improving in such a way connectivity of the combined entities. The nature of this resulting structure is of heterogeneous origin, i.e. the edges in different layers are of different type. To represent it, I employ multi-layer networks, which introduces additional challenges. This makes the straightforward use of most existing link prediction methods impossible. Current solutions for bipartite multi-layer networks oriented to drug-target interaction prediction are limited by the number or type of layers, e.g. three networks, with two assumed to be similarity networks [28, 34, 102]. In this thesis, I develop solutions for solving this.

### Modern drug development

Development of new drugs is a very important and expensive process in modern pharmacology. The process is complex and consists of multiple repetitive steps. Any mistake in any of these steps might result in a waste of resources: hours of work of field's experts and money invested into the research.

The whole process of creating a new drug to treat a given therapeutic site or disease can be described as a several step progress. In the first step, search for candidate compounds is performed. A molecule cannot be picked by random from the set of chemical elements and optimized until it can be used as a drug. The probability of successful outcome of this event tends to be zero, because the search space of chemical elements and their combinations is enormous. To develop a valid drug for a certain biological target, many molecules must be tested first to find a subset of active ones towards a given target. For this step, high-throughput screening (HTS)[6] is performed to screen the candidate compounds in order to find ones showing activity regarding a therapeutic target. In modern drug development, this search process therefore has to be automated. That is why computational methods are employed more and more frequently, while the main approach to reliably identifying such molecules still depends on *in vitro* testing. Computational methods, in turn, are able to fine-tune the set of candidates *in silico*, cutting down on time and money invested in real-world testing.

Despite the rise of computational approaches, HTS remains an indispensable part of the process. The screening technology produces a large amount of chemoinformatics data presented as set of high-throughput tests of molecules which might be used as a drug, and the next step is about processing of this information. As an example of these data, there are sets of molecules which must be classified as promising or not [9]. A molecule is understood as promising when it is active towards a given target or it can be modified in such a way that it will interact with a particular target. A molecule is not promising when it does not interact with a given target or this modification is not possible, and thus it cannot be used in the following drug development and must be excluded from consideration. A problem can appear during this step. Depending on the test readout during HTS, certain molecules can emerge as active towards a given target that do not actually interact with the target. Such false positives are impossible to optimize for the therapeutic target and therefore do not lead to successful drug development. This problem of identification and elimination of such promiscuous molecules is a separate important problem of the drug development process.

Once such a subset of valid candidates is found, they pass to the following step where they will be tested more carefully to find ones that might be modified to lead to a drug, i.e. their activity, stability, absorption *etc.* might be improved. The full drug development process includes several runs of such analysis each of which is time consuming and expensive in terms of money. At the end of this work, the drug can go to the last step, clinical testing, after passing which it goes to the market. In the negative case it cannot go to clinical testing and thus to the market, which means a lot of time and money spent for the development will be wasted.

The application side of this thesis is mostly focused on the problem of searching of drug candidates for a given therapeutic target formulated in the form of drug-target interaction prediction.

However, the problem of identification and characterization of promiscuous compounds, mentioned above, is also addressed in this thesis as an additional research question.

## Contributions

This thesis addresses two problems: 1) link prediction in bipartite multi-layer networks in general, and 2) drug-target interaction prediction as an application of the first. The motivation for the first comes from the second: modeling the problem of drug-target interaction prediction as link prediction in a bipartite multi-layer graph helps to fit data coming from different origin together and to improve the quality of prediction. State of the art solutions have some limitations regards the number and nature of networks. I introduce several approaches, which are able to solve the given problem and are free from those limitations. In addition, I propose two solutions on how to improve the computational drug development process.

In this thesis, I answer the following research questions:

1. Is it possible to develop an approach for link prediction in bipartite multi-layer networks without the limitations of the state of the art (without fixing the number and the types of layers)?
2. Does adding networks actually help to improve the quality of drug-target activity prediction?
3. Is it possible to evaluate predicted interactions "in silico" (i.e. without real-life experiments performed by a chemist), for instance on an external knowledge source?
4. Are the link prediction approaches that I develop only working with biological data?
5. (additional question) Is it possible to identify and characterize promiscuous compounds using machine learning and data mining to be able to exclude them from the drug development process at early stage?

The contribution of the thesis to answer question 1 is the following:

- I propose the NEWERMINE (NEtWork-basEd Random walk on MultI-layered NEtwork) algorithm, based on matrix multiplications for solving the given link prediction problem. Compared to the state of the art solutions, my approach is not dependent on fixing the number and type of networks. This contribution is published in the IDA'18 conference proceedings.
- To improve the performance of the preceding algorithm, I propose the LPBYCD (Link-Prediction-by-Community-Detection) approach, which holds only one bipartite layer (without limiting the number of other layers), but provides better results and is able to auto-tune its parameters. This contribution is published in the SAC'20 conference proceedings.

The contribution of the thesis towards question 2 is the following:

- To perform the experiments mentioned above, I constructed multi-layer network consisting of six layers. Taking into account that available bipartite multi-layer networks are rare, that network, which I constructed, referred to as the IUPHAR network, is a contribution itself. This network is described in the IDA'18 conference publication.
- I perform extensive evaluation of NEWERMINE on the IUPHAR network and other publicly available data. I explain why and in which cases adding networks helps to improve the quality of link prediction. I demonstrate experimentally that adding networks helps to improve connectivity of data and that optimizing certain of these characteristics improves the quality of link prediction. These results have been presented at GEM'19 workshop as part of the ECML-PKDD'19 conference.

The contribution of the thesis towards question 3 is the following:

- I develop a framework for the validation of drug-target interaction prediction founded on an external resource. The experimental results show that it is possible to use this framework

to validate predictions made on unseen data. This framework is mentioned in the SAC'20 conference publication.

The contribution of the thesis towards question 4 is the following:

- Using publicly available data, I construct two bipartite multi-layered data sets: the MovieLens network of user-movie recommendations and the Unicode languages network of country-language interactions. The origin of these sets are different than drug-target interactions, and I thus refer them to as generic. Taking into account that data for the setting that I work on are rarely available, those networks themselves can be considered as contribution.
- The evaluation of the most recent LPBYCD approach on generic data sets demonstrate that my approach is able to perform link prediction in a bipartite multi-layer network regardless of its origin.

The contribution of the thesis towards question 5 is the following:

- I develop an approach that is based on graph mining, sampling and machine learning for identification and characterization of such promiscuous compounds. In addition to the approach, I develop a graphical interface for a chemical expert for performing predictions and the result exploration. This contribution is published in the KDD'18 conference proceedings, and the interface demo is presented at the ECML-PKDD'20 conference.

## Structure of the thesis

This thesis is organized into four parts. Part I provides the foundations for the rest of the thesis. Parts II and III are dedicated to link prediction in the bipartite multi-layer graph setting problem: part II presents the random walker based approaches and the latter discusses how community detection can be used to perform link prediction. The last, part IV, is dedicated to the identification and characterization of promiscuous compounds problem.

In more details, the thesis is organized as the following.

**Part I** Chapter 1 provides the basic concepts and definitions used through the rest of the manuscript. Chapter 2 describes the data used for experiments and the construction of new data sets for the given setting, including the IUPHAR network and the generic data sets. Chapter 3 presents the framework for validation of predicted drug-target interactions founded on an external resource.

**Part II** Chapter 4 presents the EXPLICIT RANDOM WALKER algorithm based on the classical random walk model used later as a baseline. Chapter 5 presents a more efficient approach, NEWERMINE, based on matrix multiplications. Chapter 6 provides the results and the comparison of NEWERMINE and EXPLICIT RANDOM WALKER, and the results of using the framework for validation of predicted drug-target interactions based on the external resource. Chapter 7 discusses the results of an extensive evaluation of NEWERMINE on different networks with different number of layers.

**Part III** Chapter 8 presents the LPBYCD approach. Chapter 9 provides the extensive results of LPBYCD on different data including generic sets.

**Part IV** Chapter 10 presents the approach for identification and characterization of promiscuous compounds and the graphical interface, PREPEP, for the result exploration.

I finish with the conclusion and future directions chapter, where I summarize the overall contribution of the thesis and list perspectives which can be considered as future extensions of the proposed works in this thesis. At the end, in the publications and prototypes chapter, I list the publications, in which the contributions presented in this thesis appear, and the prototypes I developed while working on this thesis.

# Preliminaries


1 **Foundations** . . . . . 7
1.1 Definitions
1.2 Problem setting

2 **Data sets** . . . . . 21
2.1 Drug-target interactions
2.2 Generic data sets

3 **Validation of link prediction founded on external knowledge** . . 31
3.1 Principle
3.2 The framework to validate link prediction
3.3 Conclusion

# 1. Foundations

*In this chapter, I provide definitions of basic concepts used in this manuscript. They are organized into three groups: concepts related to chemoinformatics, graph definitions and prediction quality metrics. I then review the literature on the drug-target prediction problem. Next, I explain motivation of using the bipartite multi-layer graph setting and present the state of the art for that setting. At the end, I formally define the problem that will be addressed in the following chapters of this thesis.*

## 1.1 Definitions

In this section, I provide definitions of all the basic concepts used in this thesis. I start with the chemical and biological definitions grouped into the chemoinformatic concepts subsection, then I present definitions related to graphs, and I finish by presenting prediction quality metrics used to evaluate our approaches.

### 1.1.1 Chemoinformatic concepts

The most important concepts in our work are *ligand* and *biological target*. We understand by ligand a small molecule having organic origin that can be optimized to be used as a drug to treat a specific biological target or several targets. More precisely we define ligand as follows.

**Definition 1.1.1 — Ligand.** Ligand is a chemical compound that activates or inhibits the function of a biological target or several targets.

We use the notions of *ligand* and *drug* interchangeably in this manuscript, because of their similar semantics. By biological target we understand a molecule having biological origin that is responsible for a disease, e.g. part of a virus or bacteria, also a place in the human body responsible for the symptoms which need to be treated. More precisely, biological target can be defined as follows.

**Definition 1.1.2 — Biological target.** Biological target or just target is a protein that a ligand or a drug can bind to resulting in a change of protein behavior or function.

For simplicity reasons, we use the notions of *target* and *protein* interchangeably, because of their similar semantics.

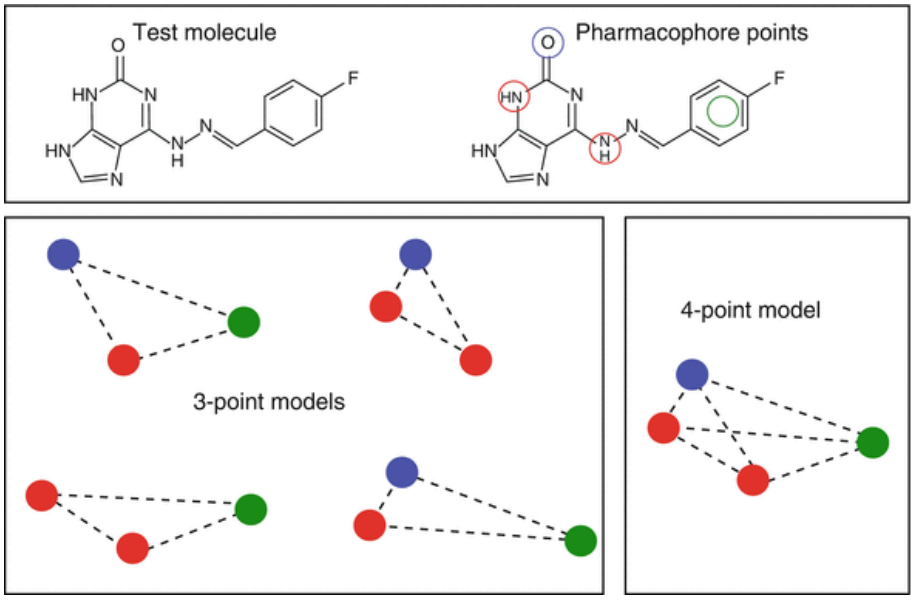


Figure 1.1: An example of pharmacophore features for a test molecule and the corresponding to them 3 and 4 point models [11]

Another biochemical concept, *protein motifs*, is used in our work to compute similarity networks. Protein motifs can be described as biochemical patterns often expressed in the form of regular expressions. More precisely, we define this concept as follows.

**Definition 1.1.3 — Protein motif.** Protein motif or just motif is an amino-acid sequence pattern that is frequent enough and has, or is presumed to have, a biological meaning.

We exploit the amino-acid sequence representation of proteins to discover the protein motifs. Then we use these motifs as features to compute the protein similarity.

Two next three concepts, *pharmacophore*, *virtual screening* and *chemogenomics*, are used in our survey on the drug-target prediction to describe the state of the art methods.

**Definition 1.1.4 — Pharmacophore.** Pharmacophore is the 3D arrangement of molecular features that are necessary for a ligand to interact with a biological target in a specific binding site [51].

A pharmacophore model searches for the largest 3D pattern of features responsible for binding that is shared by all or most given ligands (Fig. 1.1 for an example). Such a pattern can be used after to identify novel ligands that will bind to the same biological target. The latter is usually achieved by *virtual screening*.

**Definition 1.1.5 — Virtual screening.** Virtual screening is a computational technique used in drug discovery to search libraries of small molecules in order to identify those structures which are most likely to bind to a specific biological target, typically a protein [141].

Virtual screening basically evaluates large libraries of chemical compounds in an automatic way using a computer program and outputs a list of drug candidates for a given biological target.

**Definition 1.1.6 — Chemogenomics.** Chemogenomics is the systematic screening of chemical libraries of small molecules against individual drug target families (e.g., G protein-coupled receptors, nuclear receptors, kinases, *etc.*) with the ultimate goal of identification of novel drugs and drug targets [24].

Chemogenomics is a fundamental principle used in modern drug discovery for searching new drug-target candidate pairs.

### 1.1.2 Graph definitions

Graph is an essential structure in this thesis. We define what is graph through the definition of *labeled graph*.

**Definition 1.1.7 — Labeled graph.** We define a labeled graph as a tuple $< V, E, \lambda_v, \lambda_e >$, where $V = \{v_1, v_2, ..., v_n\}$ denotes a set of vertices, $E \subseteq V \times V$ a set of edges defined by distinct vertex pairs $(u, v) \in V \times V$ with $u \neq v$ (without self-loops) and $\lambda_v : V \mapsto \mathscr{A}_v$ a labeling function mapping vertices to elements of an alphabet $\mathscr{A}_v$ of possible vertex labels, and $\lambda_e : E \mapsto \mathscr{A}_e$ a labeling function for edges.

We exploit this representation in two ways. First, drugs are represented by their *molecular 2D-structure*, with $\mathscr{A}_v$ a subset of atoms, and $\mathscr{A}_e = \{$single covalent bond, double covalent bond, triple covalent bond$\}$. For example, *Pyridin-4-amine* has chemical equation $C_5H_6N_2$ and can be presented as in Fig. 1.2. Second, the relationships between drugs, targets, or between drugs and targets, are represented as *networks*, in which $\mathscr{A}_v$ is the set of drug/target identifiers (Fig. 1.3). Note that in this thesis, we use terms *node* and *vertex* interchangeably, also an *edge* sometimes refer to as a *connection* or a *link*.

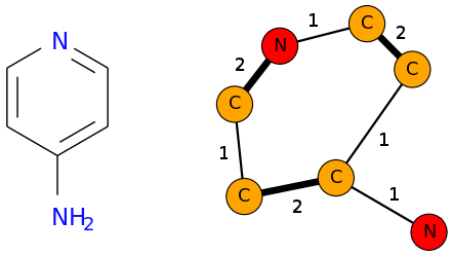


Figure 1.2: Example of a molecule 2D representation (on the left) and its corresponding graph in hydrogen suppressed form (on the right)

We address *weighted* and *unweighted* graphs in the same manner.

**Definition 1.1.8 — Weighted graph.** We define a weighted graph as labeled graph $G =< V, E, \lambda_v, \lambda_e >$ with a labeling function for edges $\lambda_e : E \mapsto \mathscr{A}_e$ with $\mathscr{A}_e \in [0, 1]$, value of which represents a link probability (in case of an interaction network) or vertex similarity (in case of a similarity network).

**Definition 1.1.9 — Unweighted graph.** We define an unweighted graph as a weighted graph every edge of which is labeled by 1: $< G = V, E, \lambda_v, \lambda_e >$, with $\lambda_e : \mathscr{A}_e = \{1\}$.

We also use the notion of a *bipartite graph*, which we define as follows.

**Definition 1.1.10 — Bipartite graph.** We define bipartite graph as a labeled graph whose vertices can be divided into two classes $V_1$ and $V_2$ such that there is no edge between vertices of the same class: $G =< V_1 \cup V_2, E, \lambda_v, \lambda_e >, E \subset V_1 \times V_2$.

To exploit different sources of information in one single structure, we employ multi-layer networks.

**Definition 1.1.11 — Multi-layer network.** We define multi-layer network as a weighted graph $G = G^{(n)}$, where $G^i = \langle V, E^i, \lambda_v, \lambda_{e^i} \rangle$ is layer i out of n layers of $G$, $\forall (u, v) \in E$, $u \in V, v \in V$

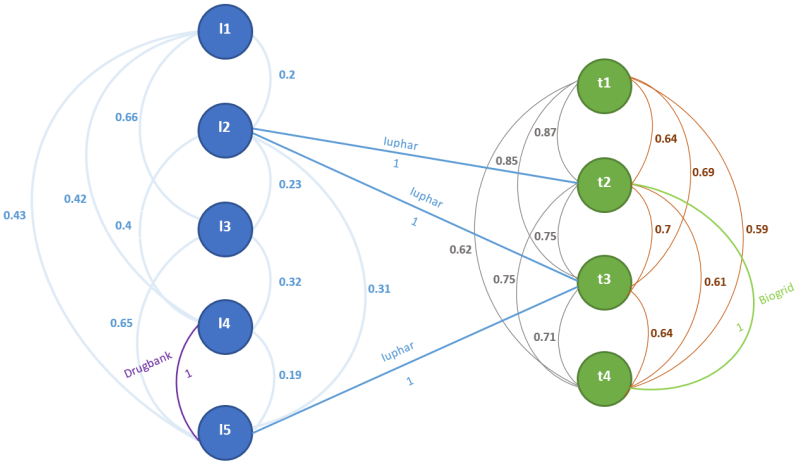


Figure 1.3: An example of a bipartite multi-layer graph with one bipartite layer and five additional layers represented by different colors: the bipartite layer is in deep blue (IUPHAR), drug networks are in light blue (ligand similarity network) and violet (DrugBank), target networks are in green (BioGrid), grey (target similarity network based on substrings) and brown (target similarity network based on motifs)

and $\lambda_v \mapsto \mathscr{A}_v$ as a labeling function mapping vertices to elements of an alphabet $\mathscr{A}_v$ of possible vertex labels, and $\lambda_{e^i} : E^i \mapsto [0,1]$ as a labeling function for edges.

Contrary to single layer graphs, in a multi-layer network more than one edge $(u,v)$ can exist for a pair of vertices $u,v$. Moreover, multi-layer network can be decomposed into disjunct set of graphs $G^i$ that contain at most a single edge for each pair of vertices, called *layers* or just *networks*.

As we stated in the introduction, our original setting is a bipartite one. To combine it with the multi-layer framework, we define a *bipartite multi-layer graph* as a multi-layer network whose vertices can be divided into two classes, and where exactly one of the layers is a bipartite graph (see Fig. 1.3 for an example). In a formal way we define it as follows.

**Definition 1.1.12 — Bipartite multi-layer graph.** We define bipartite multi-layer graph $G = G^{(n)}_{V_1} \cup G^{(m)}_{V_2} \cup G^{(l)}_{V_1V_2}$ with $n+m+l$ layers, where:

- $G^i_{V_1} = \langle V_1, E^i_{V_1}, \lambda_{v_{V_1}}, \lambda_{e^i_{V_1}} \rangle$ is layer i of type $V_1$ out of n layers of type $V_1$ with $u,v \in V_1$ labeled with identifiers of type $V_1$, $\forall (u,v) \in E_{V_1}, u \in V_1, v \in V_1$ and $\lambda_{v_{V_1}} \mapsto \mathscr{A}_{v_{V_1}}$ as a labeling function mapping vertices to elements of an alphabet $\mathscr{A}_{v_{V_1}}$ of possible vertex labels of types $V_1$, and $\lambda_{e^i_{V_1}} : E^i_{V_1} \mapsto [0,1]$ as a labeling function for edges,
- $G^j_{V_2} = \langle V_2, E^j_{V_2}, \lambda_{v_{V_2}}, \lambda_{e^j_{V_2}} \rangle$, is layer j of type $V_2$ out of m layers of type $V_2$ with $u,v \in V_2$ labeled with identifiers of type $V_2$, $\forall (u,v) \in E_{V_2}, u \in V_2, v \in V_2$ and $\lambda_{v_{V_2}} \mapsto \mathscr{A}_{v_{V_2}}$ as a labeling function mapping vertices to elements of an alphabet $\mathscr{A}_{v_{V_2}}$ of possible vertex labels of types $V_2$, and $\lambda_{e^j_{V_2}} : E^j_{V_2} \mapsto [0,1]$ as a labeling function for edges,
- $G^k_{V_1V_2} = \langle V_1 \cup V_2, E^k_{V_1V_2}, \lambda_{v_{V_1V_2}}, \lambda_{e^k_{V_1V_2}} \rangle$ is bipartite layer k out of l layers of type $V_1V_2$ with $u \in V_1$ labeled with identifiers of type $V_1$, $v \in V_2$ labeled with identifiers of type $V_2$, $\forall (u,v) \in E, u \in V_1, v \in V_2$ and $\lambda_{v_{V_1V_2}} : V \mapsto \mathscr{A}_{v_{V_1V_2}}$ as a labeling function mapping vertices to elements of an alphabet $\mathscr{A}_{v_{V_1V_2}}$ of possible vertex labels of both types $V_1$ and $V_2$, and $\lambda_{e^k_{V_1V_2}} : E^k_{V_1V_2} \mapsto \{0,1\}$ as a labeling function for edges.

Note, following definitions of Kivelä *et al.* [91], the multi-layer networks used in this manuscript are **not** *node-aligned*[1], **not** *layer-disjoint*[2], have *diagonal couplings*[3] which are *categorical*[4], and the number of layers can be any.

According to Def. 1.1.12, an arbitrary number of networks can be aggregated into a multi-layer graph, only if their vertices can fit into one of the three categories: $V_1$, $V_2$ or $V_1 \cup V_2$. Intuitively, inside one these categories, even networks which contain a subset of vertices can be aggregated, by adding missing vertices and without adding edges between them. However, for bipartite layers, i.e. the case of $V_1 \cup V_2$, this is not really useful in practice, because such vertices do not have any edges in the networks from which they are missing.

The basic properties of a graph include the number of vertices $|V|$ and the number of edges $|E|$. We use them to define *sparsity*, another important metric widely used in this thesis. There are multiple definitions of the sparsity exist, we use one defined through the edge density [65].

**Definition 1.1.13 — Sparsity of a graph.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, we define the sparsity of $G$ as the ratio of the number of existing edges to the maximum possible number of edges:

$$Sparsity(G) = \frac{2|E|}{|V|(|V|-1)}.$$

The value of sparisty indicates how spare the network is: if the value is close to 0, the network is weakly connected, and if it is close to 1, the network is almost fully connected. We use sparsity to assess the quality of networks. Note, sparsity of a multi-layer network might exceed the value of 1 since each additional layer can add some edges to the network.

To manipulate biological networks, we need to represent them. In the following, we detail the representation that we used.

We mainly represent graphs by matrices, and the *adjacency matrix* is among the most important matrices used in this thesis.

**Definition 1.1.14 — Adjacency matrix of a graph.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, the adjacency matrix $A$ of $G$ has size $|V| \times |V|$, and each element $A_{ij}$ of this matrix represents the weight of the edge $(v_i, v_j)$ in case of a single-layer graph: $A_{ij} = \lambda_e((v_i, v_j))$, or the sum of the weights of multiple edges between $v_i$ and $v_j$ in case of a multi-layer network: $A_{ij} = \sum_{(v_i,v_j)\in E} \lambda_e((v_i, v_j))$.

Note that $A$ has zeros on the main diagonal, because graphs as used in this work have no self-loops (see Eq. 1.1 for an example of the adjacency matrix for the network from Fig. 1.3).

$$A = \begin{bmatrix} 0 & 0.2 & 0.66 & 0.42 & 0.43 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.23 & 0.4 & 0.31 & 0 & 1 & 1 & 0 \\ 0.66 & 0.23 & 0 & 0.32 & 0.65 & 0 & 0 & 0 & 0 \\ 0.42 & 0.4 & 0.32 & 0 & 1.19 & 0 & 0 & 0 & 0 \\ 0.43 & 0.31 & 0.65 & 1.19 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.51 & 1.54 & 1.21 \\ 0 & 1 & 0 & 0 & 0 & 1.51 & 0 & 1.45 & 2.36 \\ 0 & 1 & 0 & 0 & 1 & 1.54 & 1.45 & 0 & 1.35 \\ 0 & 0 & 0 & 0 & 0 & 1.21 & 2.36 & 1.35 & 0 \end{bmatrix} \quad (1.1)$$

[1] In node-aligned networks, all nodes are shared between all layers

[2] In layer-disjoint networks, each node is present only in a single layer

[3] Inter-layer edges, that cross layers, are only between nodes and their counterparts

[4] Diagonal couplings for which all possible inter-layer edges are present

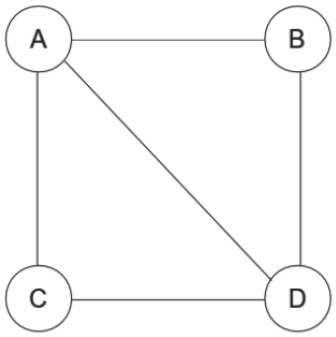


Figure 1.4: An example of a single-layer graph

Now using adjacency matrix we can define the *transition matrix*.

**Definition 1.1.15 — Transition matrix of a single graph.** Given a single graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, the transition matrix $M$ of $G$ has size $|V| \times |V|$, and in case $G$ is unweighted, each element $M_{ij}$ of $M$ has value $1/s$ if vertex $v_j$ has $s$ edges and one of them is to vertex $v_i$, and $M_{ij} = 0$ otherwise: $M_{ij} = 1/|\{(u, v_j)|(u, v_j) \in E, \exists (v_i, v_j) \in E\}|$. In case $G$ is weighted, $M = A^{norm}$, where norm denotes column-wise normalization: $\forall j \sum_i M_{ij} = 1$.

The transition matrix of a single graph is equal to its column-normalized adjacency matrix, the entries of each column of which sum to 1 (see Eq. 1.2 for an example of the transition matrix for a single-layer graph shown in Fig. 1.4).

$$M = \begin{bmatrix} 0 & 1/2 & 1/2 & 1/3 \\ 1/3 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 1/3 \\ 1/3 & 1/2 & 1/2 & 0 \end{bmatrix} \tag{1.2}$$

For multi-layer networks, there is no unified definition, and the transition matrix is defined for each concrete setting individually (see Section 5.1.4, 5.2.1 for examples).

Another matrix used in this thesis is the degree matrix, but to define it we need to define *degree of a vertex* first. We use definition from [58] for that.

**Definition 1.1.16 — Degree of a vertex.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, we define the degree of a vertex as the sum of the weights of the edges adjacent to this vertex:

$$deg(v) = \sum_{u \in V, (v,u) \in E} \lambda_e((u, v)).$$

Now we can define the *degree matrix*.

**Definition 1.1.17 — Degree matrix of a graph.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, the degree matrix $D$ of $G$ is the diagonal matrix of the same size as adjacency matrix of $G$:

$$D = \begin{bmatrix} deg(v_1) & 0 & \dots \\ 0 & \ddots & 0 \\ 0 & & deg(v_n) \end{bmatrix},$$

where $deg(v_i)$ represents the degree of vertex $v_i$.

An example of the degree matrix for the same network from Fig. 1.3 as before is shown in Eq. 1.3.

$$D = \begin{bmatrix} 1.71 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.86 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.58 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4.26 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6.32 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6.34 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.92 \end{bmatrix} \tag{1.3}$$

The last important matrix used in this thesis is the *Laplacian matrix*. It is mainly used in *spectral methods* applied in this thesis for *community detection*, which is introduced later in this section.

**Definition 1.1.18 — Laplacian matrix of a graph.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, the Laplacian matrix, denoted by $L$, is a matrix of the same size as its adjacency and degree matrices $A$ and $D$, defined as the difference between $D$ and $A$: $L = D - A$.

The Laplacian matrix has the same values as the degree matrix on the diagonal, and off the diagonal $L_{ij}$ is equal to $-A_{ij}$ (see Eq. 1.4 for an example of the Laplacian matrix for the network from Fig. 1.3).

$$L = \begin{bmatrix} 1.71 & -0.2 & -0.66 & -0.42 & -0.43 & 0 & 0 & 0 & 0 \\ -0.2 & 3.14 & -0.23 & -0.4 & -0.31 & 0 & -1 & -1 & 0 \\ -0.66 & -0.23 & 1.86 & -0.32 & -0.65 & 0 & 0 & 0 & 0 \\ -0.42 & -0.4 & -0.32 & 2.33 & -1.19 & 0 & 0 & 0 & 0 \\ -0.43 & -0.31 & -0.65 & -1.19 & 3.58 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4.26 & -1.51 & -1.54 & -1.21 \\ 0 & -1 & 0 & 0 & 0 & -1.51 & 6.32 & -1.45 & -2.36 \\ 0 & -1 & 0 & 0 & -1 & -1.54 & -1.45 & 6.34 & -1.35 \\ 0 & 0 & 0 & 0 & 0 & -1.21 & -2.36 & -1.35 & 4.92 \end{bmatrix} \tag{1.4}$$

Another two concepts used in this work are *subgraph* and *graph isomorphism*.

**Definition 1.1.19 — Graph isomorphism.** Given two graphs $G_1 = \langle V_{G_1}, E_{G_1}, \lambda_{v_{G_1}}, \lambda_{e_{G_1}} \rangle$ and $G_2 = \langle V_{G_2}, E_{G_2}, \lambda_{v_{G_2}}, \lambda_{e_{G_2}} \rangle$, we call them isomorphic ($G_1 \simeq G_2$) iff all the nodes and edges of one can be mapped to another one and vice versa: $V_{G_1} \leftrightarrow V_{G_2}$, $E_{G_1} \leftrightarrow E_{G_2}$, $\lambda_{v_{G_1}} \leftrightarrow \lambda_{v_{G_2}}$, $\lambda_{e_{G_1}} \leftrightarrow \lambda_{e_{G_2}}$.

In the case of labeled and weighted graphs, all the node labels and edge weights must remain the same.

**Definition 1.1.20 — Subgraph.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, we define a subgraph $G' = \langle V', E', \lambda_{v'}, \lambda_{e'} \rangle$ as a part of labeled graph $G$ iff $V' \subseteq V \wedge E' \subseteq E \wedge \forall v \in V' : \lambda_{v'}(v) = \lambda_v(v) \wedge \forall e \in E' : \lambda_{e'}(e) = \lambda_e(e)$. We also say that $G'$ matches $G$ ($G' \preceq G$).

A graph can be isomorphic to a subgraph of another graph (Fig. 1.5). We exploit this property to build similarity networks for drugs.

Now using the definition of subgraph we can define *cover* and *support* of a graph.

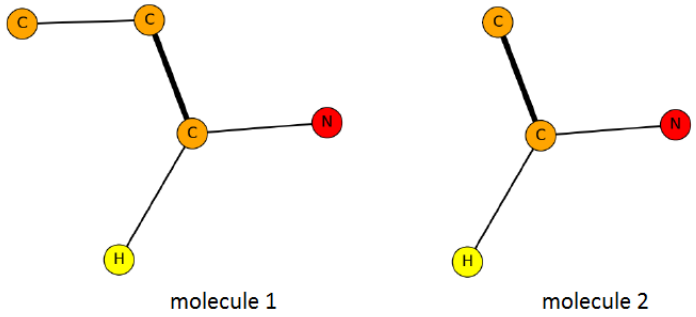


Figure 1.5: Graph of molecule 2 is isomorphic to a subgraph existing in graph of molecule 1

**Definition 1.1.21 — Cover and support of a graph.** Given a set of graphs $\mathscr{G}$, also two graphs $G = \langle V, E, \lambda_v, \lambda_e \rangle$ and $G' = \langle V', E', \lambda_{v'}, \lambda_{e'} \rangle$ such that $G' \preceq G$, we define the cover of $G'$: $cov(G', \mathscr{G}) = \{G \in \mathscr{G} \mid G' \preceq G\}$ its support: $supp(G', \mathscr{G}) = |cov(G', \mathscr{G})|$.

We use both of these notions in the description of our method for solving the last research question about identification and characterization of promiscuous compounds, mentioned in the introduction.

Using the definition of subgraph we can also define the notion of *connected component*.

**Definition 1.1.22 — Connected component.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, we define a subgraph $G' = \langle V', E', \lambda_v, \lambda_e \rangle$, $V' \subseteq V, E' \subseteq E$ as a connected component (CC) for $G$ iff for any two vertices $u, v \in V$, there exists a path $\{(v_1, v_2), \ldots, (v_{m-1}, v_m)\}, v_i \in V, (v_i, v_{i+1}) \in E$, such that $v_1 = u, v_m = v$ and there is no supergraph of $G'$, $G'' = \langle V'', E'', \lambda_v, \lambda_e \rangle, V'' \supset V', E'' \supset E'$ that is a CC. We also call $G$ is connected iff its number of CC is 1.

Obviously, to predict edges in a graph using some traditional approaches such as random walk, the network must be connected into a single component (Fig. 1.6), otherwise it becomes impossible to find a path between any two vertices in the graph. The presence of more than one connected component introduces additional challenges into the prediction process. This is why this notion is so important in link prediction, and why it can be important to evaluate that property before running predictions.

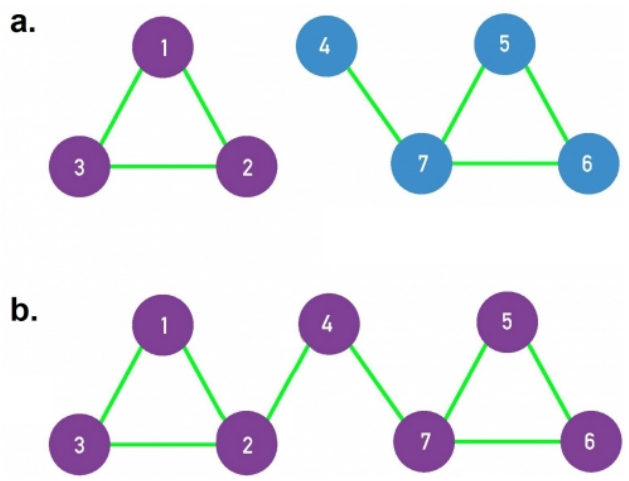


Figure 1.6: An example of a disconnected network with two *CC* shown by different colors (a) and a fully-connected network with one *CC* (b) [12]

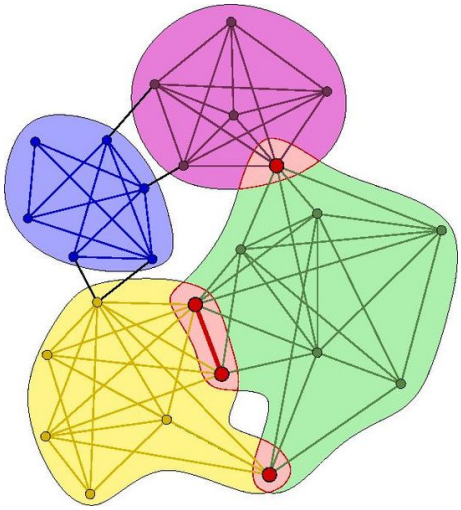

Figure 1.7: Example of non-overlapping and overlapping communities [98]

The notion of subgraph is also used to explain what *community* is, the last, but not least important notion related to graphs used in this work.

**Definition 1.1.23 — Community.** Given a graph $G = \langle V, E, \lambda_v, \lambda_e \rangle$, and it subgraph $G' = \langle V', E', \lambda_v, \lambda_e \rangle$, $V' \subseteq V, E' \subseteq E$, we refer to graph's internal degree $deg_{int} = \sum_{v \in V', u \in V', (v,u) \in E} deg(v)$, and its external degree $deg_{ext} = \sum_{v \in V', u \in V \setminus V', (v,u) \in E} deg(v)$. For $G'$ to be a community, we expect $deg_{int} \gg deg_{ext}$.

Communities are locally dense connected subgraphs in a network. Nodes in a community are more likely to connect to other members of the same community than to nodes in other communities. The blue, the purple, the green and the yellow groups of nodes in Fig. 1.7 satisfy this expectation.

To give some examples, communities could represent circles of friends, or a group of people having the same hobby together, or individuals living in the same area. From an application point of view, communities are widely used in social network analysis to predict friendship connections between people or to recommend some groups for users [61]. Communities are also used in biological networks to identify functional groups of molecules in molecule interaction networks [137] or to find proteins involved in the same disease in protein interaction networks [63, 115].

Note that, in contract to groups produced by traditional clustering approaches, communities can overlap (the purple, the green and the yellow groups in Fig. 1.7 are *overlapping* communities). Due to the nature of community detection algorithms used in this thesis, we deal with *non-overlapping* communities only (the blue group in Fig. 1.7).

### 1.1.3 Prediction quality metrics

In this thesis, we develop methods to solve the *binary classification task*.

**Definition 1.1.24 — Binary classification task.** The binary classification task is the task of classifying the elements of a given set into two groups of positive and negative elements on the basis of a prediction method.

To evaluate our methods we use common performance measures: *accuracy*, *precision* and *recall*.

**Definition 1.1.25 — Accuracy.** We define accuracy as the ratio of examples correctly classified

as positives and examples correctly classified as negatives over all predictions:

$$Acc = \frac{TP+TN}{TP+FP+TN+FN},$$

where TP – true positives (examples correctly classified as positives), TN – true negatives (examples correctly classified as negatives), FP – false positives (examples incorrectly classified as positives) and FN – false negatives (examples incorrectly classified as negatives).

**Definition 1.1.26 — Precision.** We define precision as the ratio of examples correctly classified as positives over all examples classified as positives:

$$Prec = \frac{TP}{TP+FP}.$$

Precision measures whether a model is specific enough to mainly classify links of the positive class as positive.

**Definition 1.1.27 — Recall.** We define recall or true positive rate as the ratio of examples correctly classified as positives over all positive examples in the test data:

$$Rec = \frac{TP}{TP+FN}.$$

Recall measures whether a model is general enough to classify a large proportion of the positive class as positive.

In addition to measures presented below, we also use *area under receiver operating curve* (AUC) and *area under precision-recall curve* (AUPR).

**Definition 1.1.28 — Receiver operating curve.** We define the receiver operating curve (ROC-curve) as the curve created by plotting the true positive rate against the false positive rate at various threshold settings, with false positive rate (FPR) defined as:

$$FPR = \frac{FP}{FP+TN}.$$

AUC evaluates whether true positives are usually ranked above or below false positives when sorting predictions by confidence [54].

**Definition 1.1.29 — Precision-recall curve.** We define the precision-recall curve (PR-curve) as the curve created by plotting precision against the true positive rate at various threshold settings.

AUPR summarizes the trade-off between false negative and false positive rates.

## 1.2 Problem setting

In this section, I survey the literature on drug-target interaction prediction focusing on a variety of settings and techniques, and their challenges. I then present the state of the art about link prediction in bipartite multi-layer graphs with relation to the task of drug-target interaction prediction and explain the motivation of using this setting. At the end, I define the problem of drug-target interaction prediction in a bipartite multi-layer graph in a formal way.

### 1.2.1 Survey on drug-target prediction

A large number of computational approaches has been developed since the problem of drug-target interaction prediction was first introduced. We group them into the three categories: traditional approaches, chemogenomic approaches and multi-layer network based.

#### Traditional approaches

Classical approaches perform search of valid drug candidates for a selected target (also called *virtual screening*) or valid target candidates for a fixed drug, depending on how the problem is formalized. The first is usually solved by ***molecular docking***, when the compounds are virtually screened toward target proteins [36, 39, 99, 120, 135]. This group of approaches use 3D structures of a drug and a protein and run a simulation process to determine whether they would interact. The main drawback of them is that the 3D structure of the majority of drugs is difficult to obtain, and there are proteins for which the 3D structure is not known, so that docking cannot be applied to them [53].

Another group of approaches that search valid drug candidates for a given target use ***chemical similarity models***. Based on the assumption that similar drugs interact with similar targets, they exploit 2D structure similarities of drugs to infer new drug-target interactions [33, 89]. However, similarity approaches are not applicable in situations when no known drug interactions for a given target exist (or they are simply not tested yet), because in that case comparison is not possible [92]. The other type of classical approaches to virtually screen the drug candidates include ***pharmacophore modeling and QSAR analysis***, usually limiting the task to more specific setting e.g. searching for a particular type of drug or interaction [5, 155, 175].

The problem can be formulated another way around as searching for valid targets for a given drug. One way to solve the problem is with ***bioactivity profile search***. The method is based on the same assumption as before, that similar drugs interact with similar targets, but now similarity is computed using bioactivity profiles from publicly available databases describing behavior of selected drugs across multiple targets [38, 77, 161]. An alternative way to solve the problem is by ***combining data mining with machine learning*** approaches. The first is used to mine features to represent drugs and the latter to solve the binary classification task. The training is performed on confirmed drug-target interactions, and many machine learning algorithms from Naïve Bayes to Support Vector Machines are used as a classifier [14, 124, 132, 162].

#### Chemogenomic approaches

Traditional approaches do not take into account that existing drugs might be used to treat diseases different from those they were initially developed for. The rise of *drug repositioning*, a common strategy for drug discovery that significantly reduces the cost and speeds up the discovery process, created another subset of *chemogenomic* approaches. Chemogenomic approaches use information from both the drug and target sites at the same time to perform prediction of new interactions without focusing on concrete drugs or targets [24, 53]. The straightforward way of solving this is the ***Bipartite local models*** (BLM) method, where predictions are performed for each drug and each target separately using one of the strategies discussed before [19, 166]. Once this step is performed, an average estimation for a given drug-target pair is computed between the two independent predictions. An alternative way to solve the problem is by ***neighborhood based methods*** using nearest profile or weighted profile functions [160, 170] or in combination with BLM models [114]. The main challenge of this group of approaches is prediction of new interactions between drugs and targets without knowing any prior connections with the opposite type.

In spite of the presence of the BLM models and neighborhood based methods, the most common group of methods in chemogenomic setting is the ***machine learning based methods***. In this group of methods, drugs and targets are represented by features and their interactions as one of binary classes. An estimation of a new interaction is determined by prediction of its class label by using a

machine learning classifier. The problem is solved using a number of existing machine learning techniques [31, 76, 83, 177]. These methods are usually dependent on prior knowledge of drugs and targets for constructing features. In addition, selection of negative examples is a common problem of machine learning based approaches. Negative samples are often not available, since biomedical literature publishes only successful interactions based on biological experiments [35]. In this situation, unknown interactions are regarded to be negative examples, which might negatively influence prediction accuracy. Moreover, the number of experimentally verified positive samples in such data is relatively small, what creates another issue with data imbalance.

Another common problem chemogenomic settings have to deal with is sparsity of data. As was noted, the number of experimentally validated positive interactions is usually small. Whether it is because a target has only recently been identified, because a disease is rare and commercially unattractive, or because the relation between certain compounds and targets has not been evaluated for any biological reasons. As a result the large space of possible links remains under-explored. A similar problem setting that also faces the sparsity problem is that of product recommendation. A simple ***recommendation system*** approach implements, for instance, the reasoning that if two drugs are both linked with several shared targets, and one of them is linked to an additional one, it is reasonable to assume that the other one should be linked as well. Such an approach has the advantage of exploiting information that is not directly linked to the chosen target yet is still faced with a sparsity problem since, as mentioned, most drugs do not have enough interactions to start with. This makes some recommendation system approaches for large-scale prediction, such as collaborative filtering [102] or matrix factorization [64, 105], difficult or impossible to use.

One relatively recent proposal to address sparsity problem consists of ***using network data*** where vertices represent drugs and targets, and edges between them their interactions [28, 34, 37, 49]. While this does not solve the sparsity problem by itself, it allows to introduce additional information (such as drug or target similarity constructed from their properties, or interactions between drugs or targets mined from the literature) in order to enrich available data and make it better connected. This idea needs to answer two questions: which information sources should one use and how can different networks be integrated? *In this thesis, we therefore propose to use a multi-layer network to solve this problem.*

### Multi-layer network based approaches

Existing methods for link prediction in multi-layer networks for addressing the drug-target interaction problem can be grouped into three classes: similarity based, random-walker based and latent models based.

***Similarity based*** methods assume two out of three possible layers to be similarity networks for drugs and targets respectively, and exploit similarity information to perform link prediction on the third bipartite layer [28, 49]. The next group of methods, ***random-walker based***, models the behavior of a random-walker to perform link prediction in a multi-layer graph using matrix formulations and PageRank adaptations [34, 37]. Such methods are dependent on fixing the number of similarity networks and might have high computational cost (depending on additional techniques used to improve predictions).

Finally, the ***latent model based*** methods map drugs, targets and their interactions into a combined feature space, and perform drug-target interaction prediction using distance functions or regression analysis [170, 171, 178]. The most recent family of methods in this mold is often referred to as graph embeddings [66]. The main disadvantage of methods in this group is a certain lack of interpretability.

In this thesis, we try to challenge the limitations which the state of the art approaches discussed in this section have, by developing our own methods.

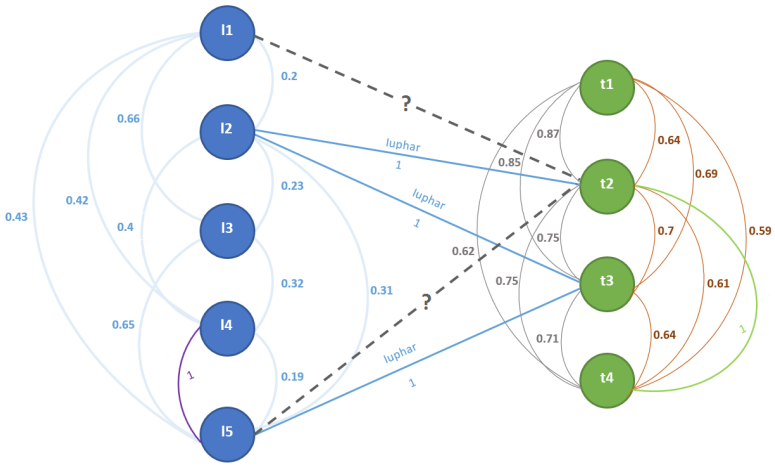


Figure 1.8: An example of activity prediction in a bipartite multi-layer graph with six layers. New possible interactions are presented by dashed edges with question marks

### 1.2.2 Problem definition

As mentioned in Section 1.2.1, data used for large scale drug-target prediction with chemogenomic setting create sparsity problems. We believe that modeling the problem of drug-target interaction prediction as a link prediction problem in a graph should help to overcome it.

The idea of collecting and organizing all sources of information into a graph representation should decrease sparsity on the entire network, however it raises additional questions of which information sources to use and how. Instead of picking and choosing between different sources of information, we propose to use all of them, exploiting different similarity and interaction information.

Considering the bipartite nature of the task, we define the problem of drug-target interaction prediction as a problem of *link prediction in bipartite multi-layer graph*. In a formal way the problem is addressed as follows.

**Definition 1.2.1 — Drug-target activity prediction.** For a given bipartite multi-layer graph $G$ defined as in Def. 1.1.12 with $V_D$ denoting set of drug vertices, $V_T$ denoting set of target vertices predict, whether for a given bipartite layer $k$ and a given pair of vertices $d_i \in V_d, t_j \in V_t$ such that $(d_i, t_j) \notin E$, edge labeling function for this pair $\lambda_{e^k_{V_D V_T}}((d_i, t_j)) = 1$.

We limit ourselves to the relatively easier task of predicting whether there is an activity or not, leaving the prediction of its *strength* as a perspective (Fig. 1.8).

# 2. Data sets

*In this chapter, I provide a description of all data sets used in this thesis. They are organized into two groups: drug-target interaction and generic data sets. I start with a description of IUPHAR, the drug-target interaction database curated by the International Union of Basic and Clinical Pharmacology. I provide motivation and describe construction of our own network based on IUPHAR. I then present benchmark data sets used for assessing performance of drug-target interaction prediction and point out to their weak points. Finally, I present two generic sets and describe motivation and construction of the networks based on them.*

## 2.1 Drug-target interactions

As presented in the introduction, there are not many data sets related to the task of drug-target interaction prediction. I start with the description of one of these data sets, IUPHAR, and describe construction of a multi-layer network based on that. I then present the other sets, called benchmark, and list their drawbacks.

### 2.1.1 The IUPHAR network

We start with detailing the IUPHAR database, then move on the construction of a multi-layer network based on that.

#### The IUPHAR database

*IUPHAR* [129] is an open-access database of ligands, biological targets, some of their properties and interactions[1]. The version used in this thesis[2] has 8978 ligands, 2987 targets and 17198 interactions (edges) between them[3], which results in an extreme sparsity of 99.92% (only 0.08% of possible links are present). It also explains the fact that there are 443 connected components when modeling data as a graph (Fig. 2.1).

As for available properties, the database stores name, INN (International Non-proprietary Name), type, 2D chemical structure in *SMILES* format along with identifiers in different standards and references to other chemical databases (such as PubChem *etc.*) of ligands. Analogously,

[1] `http://www.guidetopharmacology.org`

[2] verion 2017.5 released 22/08/2017 was used

[3] in *ligands.csv*, *interactions.csv*, and *targets_and_families.csv* files, respectively

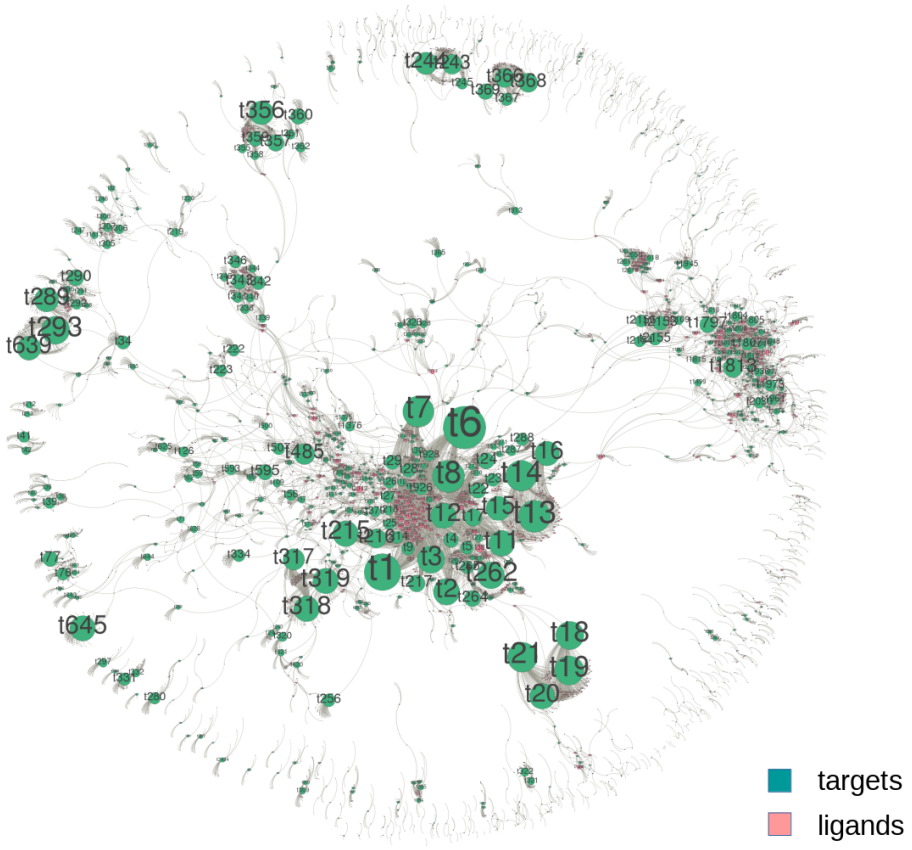


Figure 2.1: Indegree of IUPHAR's targets (ligand-target interactions are treated as a directed network, sizes of target vertices represent indegree)

targets are represented by name, class, family, different identifiers (Human Entrez Gene[4] *etc.*) and references to other biological databases (SwissProt[5] *etc.*). Concerning the interactions themselves, in addition to standard identifiers of ligands and targets two important properties are available: affinity measure type and affinity measure value.

Enormous sparsity of IUPHAR might result in a low performance of traditional recommendation system approaches. At the same time, disconnected networks are challenging for random-walk based approaches making numerous network components unreachable for a random-walker. We will try to decrease sparsity and number of connected components in the following sections by adding additional layers to the data using available properties and other databases.

### Data preparation

In order to satisfy the common setting conditions (one drug – one target – one interaction), we removed duplicate interactions (based on different affinity measures), leaving 12456 interactions in total. For existing interactions, we label an edge as *positive* if the negative logarithm of the affinity measure is $\geq 5$, and *negative* otherwise according to a cutoff proposed by our partner researchers from CERMN (Le Centre d'Etudes et de Recherche sur le Médicament de Normandie)[6]. The positive edge represents an interaction while negative means non-interacting pair. In such a way we split all interacting pairs into 11487 positives and 969 negatives. It is also worth noting that we

[4]Global Query Cross-Database Search System gene identifiers: `http://www.ncbi.nlm.nih.gov/gene`

[5]Freely accessible database of protein sequence and functional information: `http://www.uniprot.org`

[6]Drug design research center in Normandy: `http://cermn.unicaen.fr`

treat all affinity measures available in the data (pKi, pIC50, pEC50, pKd, pA2, pKB) as equivalent.

Since IUPHAR is very sparse and highly disconnected in its basic form, we try to solve it by computing similarity networks for both drugs and targets. We explain how we do it in the following.

To compute similarities of drugs, we converted SMILES codes containing chemical structure of ligands into graphs, from which we mined frequent subgraphs using the implementation of [173] to build descriptors for drugs. To represent targets we first collected amino-acid sequences from the National Center for Biotechnology Information (NCBI) Protein website[7] using SwissProt identifiers of targets. Then we built target descriptors in two ways: by mining frequent substrings in amino-acid sequences, and by mining protein motifs using Prosite database[8]. The first approach is straightforward, but the second produces biologically meaningful patterns. We use both representations since there is no prior knowledge on which one can be expected to be more useful.

#### Multi-layer network construction

To build additional layers for IUPHAR, which according to our hypothesis should improved its sparsity and vertex reacheability, we used 3 additional databases: *DrugBank* [163], *BioGrid* [150] and *NCBI Protein* [134].

DrugBank is an open-access database of drug-drug interactions. The version we used[9] has 658079 interactions of 3138 distinct drugs, 242922 of which involve 1254 distinct ligands that are present in IUPHAR. The second, BioGrid, is an open-access database of protein-protein interactions mined from a corpus of biomedical literature. The version we used[10] has 1482649 interactions of 67372 distinct proteins. Only 15410 of these interactions involve targets present in IUPHAR (1925 distinct targets). Finally, NCBI Protein was used to obtain amino acids sequences to represent targets, for which the sequences were parsed from the website of NCBI and mapped with IUPHAR[11].

Drugs were mapped between networks by numerical identifiers provided by IUPHAR as well as by INN and Common name attributes. Proteins were mapped by IUPHAR identifiers and by Human Entrez Gene attribute. As figures in the previous paragraph show, not all drugs and proteins contained in IUPHAR are available in DrugBank and BioGrid. In addition, not all proteins and drugs are annotated with molecular information so that not all entities will be connected in the similarity networks. In total we have built 6 networks:

1. the drug-target interaction network based on IUPHAR,
2. a drug-drug interaction network based on DrugBank,
3. a drug similarity network based on similarities calculated using the Tanimoto coefficient on binary vectors constructed from the presence/absence of the top 200 frequent subgraphs[12],
4. a target-target interaction network based on BioGrid,
5. a target similarity network calculated using the Tanimoto coefficient on feature vectors constructed from the presence/absence of the top 200 frequent substrings[12], and
6. a target similarity network calculated using the Tanimoto coefficient on feature vectors constructed from the presence/absence of the top 200 Prosite motifs[12].

Similarity networks' edges were labeled with labels $\in [0,1]$, interaction networks with labels $\in \{0,1\}$. To give an example, Fig. 1.3 illustrates an extract from the resulting network.

Table 2.1 shows the characteristics of the databases, and of the networks we derived from them. It is noticeable how sparse the data is (IUPHAR's sparsity is equal to 0.0002, BioGrid's to 0.0048

[7] NCBI Protein on-line database: `http://www.ncbi.nlm.nih.gov/protein`

[8] An open-access database is available at: `http://prosite.expasy.org`

[9] version 5.0.11 released 20-12-2017 was used

[10] version 3.4.154 released 25/10/2017 was used

[11] The database was accessed 20/12/2017

[12] An experimentally selected value.

Table 2.1: Database and network characteristics

| Data set | Database | | | Network* | | | |
|---|---|---|---|---|---|---|---|
| | Entities | Relations | Sparsity | $\|V\|$ | $\|E\|$ | Sparsity | CC |
| IUPHAR | 11965 | 12456 | 0.0002 | 11965 | 12456 | 0.0002 | 443 |
| DrugBank | 3138 | 658079 | 0.1337 | 1254 | 122808 | 0.1563 | 1 |
| BioGrid | 67372 | 1482649 | 0.0007 | 1898 | 8658 | 0.0048 | 11 |
| Drug similarity | 6821 | 23259610 | 1 | 6821 | 23259610 | 1 | 1 |
| NCBI Protein | 1818 | 1651653 | 1 | 1818 | 1651653 | 1 | 1 |
| All combined | – | – | – | 10639 | 26706838 | 0.4719 | 1 |

* Ignoring *isolated vertices*, vertices not connected to any of the nodes.

*etc.*), and also how this sparsity translates into disconnected parts of the network (IUPHAR results in 443 connected components and BioGrid in 11).

Obviously, with such a high number of connected components as the IUPHAR database has, it is unlikely to derive a lot of interactions using only the database. In the given setting, there are many cases where a path between selected drug and target pair does not exist, which makes it impossible to assess a possibility of an interaction between them.

As can be seen from the bottom row in Table 2.1, our hypothesis formulated at the beginning of this section holds. Combining all networks into one multi-layer graph significantly improves sparsity and creates one single connected component (the last two columns in Table 2.1). We will use this data set later to test our methods developed in parts II and III of this thesis.

### 2.1.2 Benchmark sets

In addition to the IUPHAR network, we also test our methods on the data sets introduced in [170]: *Enzyme*, *GPCR* (G-protein coupled receptors), *IC* (Ion Channels) and *NR* (Nuclear Receptors). In addition, we use the *Kinase* set [46]. These data sets have been used in prior work on drug-target interaction prediction [28, 34, 102, 178], and can be considered benchmarks.

#### Data set properties

The data in benchmark sets consist of three networks: drug similarities, target similarities and drug-target interaction (the bipartite graph). Drug similarity network is computed by the use of the SimComp score [73], and target similarity is by the Smith-Waterman score [147]. Drug-target interaction network is constructed from the KEGG BRITE [87], BRENDA [145], SuperTarget [70], and DrugBank databases. The data sets' basic properties are presented in Table 2.2.

Table 2.2: Basic properties of benchmark data sets

| Data set | Drugs | Targets | Interactions | $\|V\|$ | $\|E\|$ |
|---|---|---|---|---|---|
| Enzyme | 445 | 664 | 2926 | 1109 | 321832 |
| GPCR | 223 | 95 | 635 | 318 | 29853 |
| IC | 210 | 204 | 1476 | 414 | 44127 |
| NR | 54 | 26 | 90 | 80 | 1846 |
| Kinase | 68 | 442 | 1527 | 510 | 101266 |

Table 2.2 shows that the number of vertices and edges of benchmark sets are much smaller than that of IUPHAR (last row in Table 2.1), so they cannot be used to test whether new link prediction approaches scale. Moreover, the fact that the sets are well tested lowers the chances to predict new valid interactions based on them.

More detailed properties of benchmark data sets will be studied later in Chapter 7.

#### Data preparation

We downloaded the original data of four of the datasets (Enzyme, GPCR, IC, NR) from the Yamanishi *et al.* [170] supplementary information[13], and the fifth (Kinase) from the Davis *et al.*' [46] supplementary information[14].

The interaction network was originally presented as binary relation lists, while the similarity data were presented by matrices. We transformed the data to adjacency lists, and for the sake of (computational) convenience we mapped ligand and drug names to consecutive integers, i.e. $l0, l1, l2, \ldots, t0, t1, t2, \ldots$, the required input for our implementation.

## 2.2 Generic data sets

In the previous section, I presented data sets for solving the drug-target interaction prediction task. At the same time, I suppose that techniques developed later in Chapter 8 are more general. In other words, they can be used for any kind of bipartite multi-layer graph regardless of its origin. In order to verify that later in Chapter 9, in this section, I construct two new bipartite multi-layer data sets based on publicly available data. The first, "MovieLens", contains user-movie ratings and the second, "Unicode languages", concerns countries and languages spoken in them.

### 2.2.1 MovieLens

MovieLens data sets[15] are well-known benchmarks for recommendation systems [72]. Based on real-life surveys of individuals on existing movies, they are made for solving user-movie recommendation task [78].

#### The original set description

For minimizing evaluation complexity, we base construction of our multi-layer network on the oldest, and thus the smallest set of the MovieLens group[16]. The version we use includes information about 943 users, 1682 movies and 100000 movie ratings made by users. In the data users are presented by age (integer value), gender (binary value), occupation (one of 21 categories) and a post code (in the US or Canadian format for citizens from US and Canada respectively). Movies are described by name, release date, genre (19 multiple categories) and include links to Internet Movie Database (IMDB)[17].

#### Feature construction

To build similarity networks, we first create features to describe users and movies. To achieve that, we exploit available properties of the both types of entities.

**User descriptors** To describe users by age, we introduced five age groups for users: less then 18, between 18 and 25, between 26 and 34, between 35 and 49, and 50 and more. The first and the last usually have opposing preferences in movies, the middle groups were selected based on principle of balance (Fig. 2.2). Gender category was used as it is, and occupation as well. Post codes were converted into the exact location names they belong to using publicly available databases[18,19]. Then we binarised these locations into two groups: "big city" and "province" based on the top 100 US

[13] `http://web.kuicr.kyoto-u.ac.jp/supp/yoshi/drugtarget`

[14] `http://staff.cs.utu.fi/~aatapa/data/DrugTarget`

[15] `http://grouplens.org/datasets/movielens`

[16] 100K Dataset (containing the user ratings from 1998)

[17] A popular internet database of movies: `http://www.imdb.com`

[18] Database of ZIP codes of the USA states: `http://www.pier2pier.com/links/files/Countrystate/USA-Zip.xls`

[19] Database of post codes of Canada: `http://www.postalcodesincanada.com`

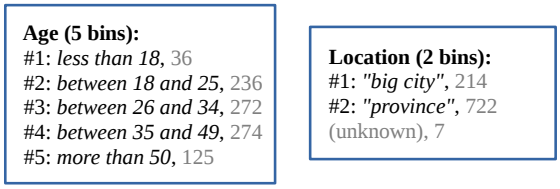


Figure 2.2: Handcrafted descriptors for users in the MovieLens data set (the number of instances in each category of each descriptor as well as the amount of unclassified instances, where it is applicable, are shown in grey color)

cities[20] and the top 10 Canadian cities[21]. The resulting partitioning produced groups with roughly 20% of individuals from "big cities" and 80% from "province" (Fig. 2.2).

To sum up, we encode each user by the vector with 30 binary values:

1. 5 bits for age (5 bins),
2. 2 bits for gender (2 categories),
3. 21 bits for occupation (21 categories),
4. 2 bits for location (2 groups).

We use 1 bit per bin because some of the properties can be unknown (they are encoded by 0s in such cases).

**Movie descriptors** The movie descriptors are more complex to build. They require more information to be merged together, however, the vectors constructed with these descriptors are more informative. The basic properties, including release date and genre, do not require much preprocessing. We only binarised the dates into 13 categories based on the principle of balance. For that we kept approximately similar number of items for each group of old movies, and for individual groups corresponding to each year of recent movies, we also used the same principle (Fig. 2.3).

IMDB identifiers provided with the data are not active anymore, and therefore we did not find a better way than to parse new IMDB identifiers from the website. We matched candidate movies by title, year and release date[22] and used edit distance algorithm [45] to filter candidates. We then processed the lists of candidates and filled in blanks manually double checking with movie genres. Once IMDB identifiers were collected, the IMDB database was parsed one more time to retrieve additional information about the movies[23]: budget, country of origin, main language, length, type (movie or other), actors, production company and crew members (directors, musicians, operators etc).

We initialized construction of additional descriptors from the budget property, which was converted from multiple currencies into US dollars using historical exchange rates[24] on 1998, 22 of April[25]. Then these values were binarised into 3 categories with meaningful ranges: less than $1 mln, between $1 mln and $10 mln, and more than $10 mln. The values of the next property, countries of origin, were placed into 5 groups using expert knowledge: USA, Britain and France are major movie production countries of all time[26], while Europe (except France) has own cinematic

[20] According to census of 1990 (published in 1998): `http://www.census.gov/population/www/documentation/twps0027/tab22.txt`

[21] According to census of 1996: `http://en.wikipedia.org/wiki/List_of_largest_Canadian_cities_by_census`

[22] The `imdb.com` was accessed on 19/10/2019

[23] The `imdb.com` was accessed again on 25/10/2019

[24] `http://www.poundsterlinglive.com/bank-of-england-spot/historical-spot-exchange-rates/usd`

[25] The last day of the MoveLens survey for 100K Dataset

[26] `http://www.the-numbers.com/movies/production-countries`

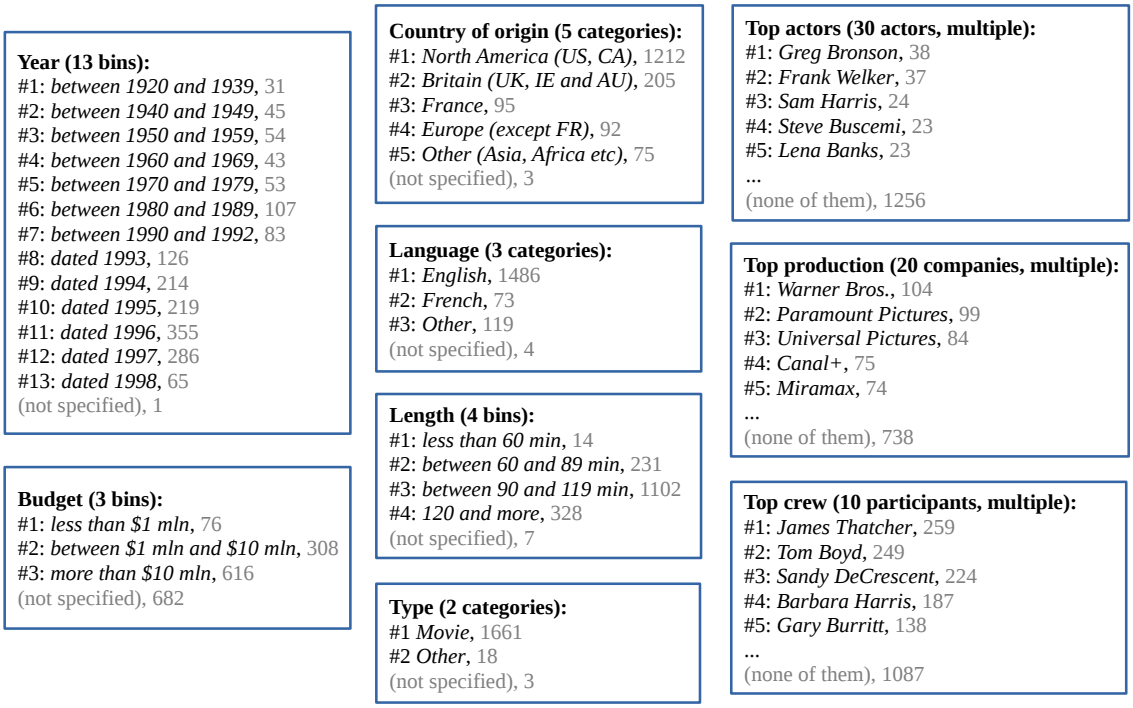


Figure 2.3: Handcrafted descriptors for movies in the MovieLens data set (the number of instances in each category of each descriptor as well as the amount of unclassified instances, where it is applicable, are shown in grey color)

trends (Fig. 2.3). We split languages into three categories: English, French and other emphasizing French movies as a separate group since France has own influence on movie production and thus can be placed separately. The lengths of the movies were binarised into four categories with meaningful ranges as shown in Fig. 2.3. Concerning the last three properties, we have chosen the top $n$ values for actors, companies and crew members allowing multiple categories. Value of $n$ for each property was selected individually using the principle of minimizing redundancy of vector representation.

To sum up, we encode each movie by the vector with 108 binary values: 18 bits for genre that goes with the data, plus 90 bits for the manually created features summarized in Fig. 2.3.

### Network construction

As in the case with IUPHAR, the construction of similarity networks is performed with the use of the Tanimoto coefficient, which is better suited for binary vector comparison. The resulting similarity values tell us about the quality of networks constructed. As can be seen from Table 2.3 (first two rows), these values are diverse enough, covering the full range from 0 to 1 with mean and median far from 0 or 1. It gives us a conclusion that the resulting similarity networks are informative enough. In other words, it would be possible to find a similar movie for a randomly selected one by maximizing their similarity value (see Fig. 2.4 for an example: *Grease 2 (1982, comedy, musical, romance)* is more similar to *Top Gun (1986, action, romance)*, than to *Star Wars (1977, action, sci-fi)*).

As defined in Section 1.2.2, we do not predict the strength of the interaction, and therefore user-movie ratings in the bipartite network need to be binarised. We use ranks from 3 to 5 to represent positive interaction and from 1 to 2 as negative. That produced 82520 positive and 17480 negative edges.

In total we have built three networks:

Table 2.3: Ranges of Generic data sets' similarity networks

| Data set | Network | Similarity values | | | |
|---|---|---|---|---|---|
| | | min | mean | median | max |
| MovieLens | User similarity | 0 | 0.27 | 0.33 | 1 |
| | Movie similarity | 0 | 0.25 | 0.24 | 1 |
| Unicodelang | Country similarity | 0.03 | 0.31 | 0.29 | 1 |
| | Language similarity | 0 | 0.31 | 0.27 | 1 |

Table 2.4: Basic properties of Generic data sets

| Data set | V1 | V2 | Interactions | $\|V\|$ | $\|E\|$ | Sparsity | CC |
|---|---|---|---|---|---|---|---|
| MovieLens | 943 | 1682 | 100000 | 2625 | 1940394 | 0.563 | 1 |
| Unicodelang | 254 | 614 | 1255 | 868 | 218996 | 0.582 | 1 |

1. the user-movie binary interaction network,
2. a user similarity network calculated using the Tanimoto coefficient, and
3. a movie similarity network also calculated using the Tanimoto coefficient.

To give an example, Fig. 2.4 illustrates an extract from the resulting network.

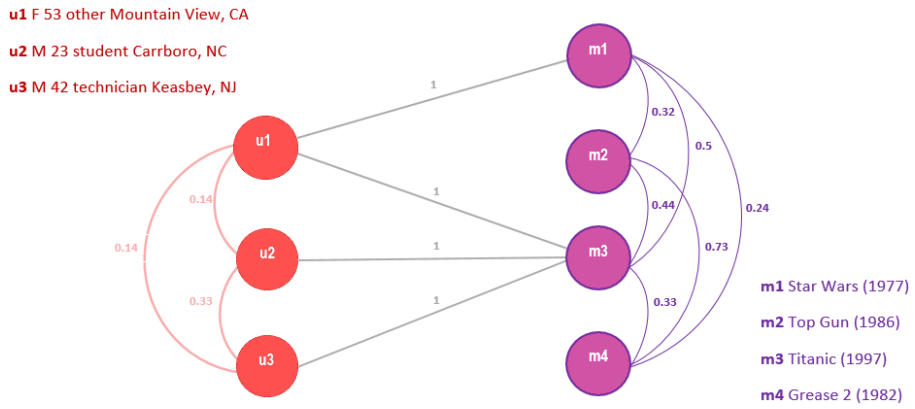


Figure 2.4: An extract from the MovieLens network with 3 layers represented by different colors: the bipartite layer is in grey (MovieLens), user similarity network is in pink and movie similarity network is in violet

Basic properties of the resulting network are presented in Table 2.4. As can be seen from this table, the MovieLens network has an acceptable sparsity and consists of a single connected component.

### 2.2.2 Unicode languages

There are not many data sets with bipartite structure which are both open source and provide a rich property set for both types. For instance in KONECT, popular open-access collection of network data sets, there is only one network satisfying these requirements [94]. Unicode languages or "Unicodelang" is that set describing languages, countries, and their relations.

#### The original set description

The Unicodelang network represents 254 countries (territories), 614 languages and 1255 of their "interactions", denoting the proportion of the population of a given country speaking a given language. As in the case with the previous data set, we perform an extension of the basic network

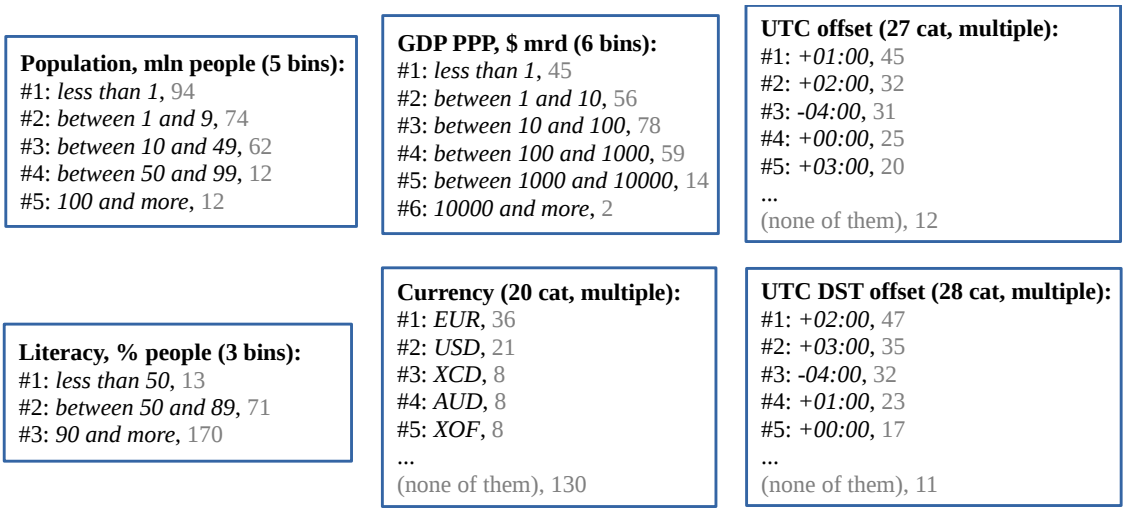


Figure 2.5: Handcrafted descriptors for countries in the Unicodelang data set (the number of instances in each category of each descriptor as well as the amount of unclassified instances, where it is applicable, are shown in grey color)

by constructing additional layers.

### Feature construction

As before, to build similarity networks we first construct features to describe both type of entities. For doing that, we parse country and language properties from the original source of the set used by KONECT. We exploit collected properties then to build descriptors for the both of the types.

**Country descriptors** The countries are presented by 10 basic properties[27]. We binarise the first three: population size, percentage of literacy and GDP PPP[28], into 5, 3 and 6 categories respectively using meaningful ranges (Fig. 2.5). Another property, currency, was placed into 20 multiple categories, ignoring distinct values and taking into account that some countries can have more than one official currency (Fig. 2.5). Next six properties we used as they are given: number of days in a week, first day of a weekend, last day of a weekend, measurement system and paper size. Next, we parse supplementary information[29] to get additional properties. That includes continent, subcontinent and two time zone offsets. The first two were used in a straightforward manner, while the time zone names of the latter were converted to the exact values using publicly available chart[30]. Then collected offset values were fitted into 27 and 28 categories for UTC and DST respectively, ignoring distinct values (Fig. 2.5).

**Language descriptors** The languages are presented by boolean properties whether a language is modern, primary or one/none of them, and a language script type, both of which we use unchanged[31]. Another property, language population, was adopted from the countries' supplementary information[29] summing up numbers for several instances (Fig. 2.6). The next two, plural rules'

[27] `http://www.unicode.org/cldr/charts/25/supplemental/territory_language_information.html`, accessed on 7/11/2019

[28] Gross domestic product based on purchasing power parity

[29] `http://www.unicode.org/cldr/charts/25/supplemental/territory_containment_un_m_49.html`, accessed on 7/11/2019

[30] `http://en.wikipedia.org/wiki/List_of_tz_database_time_zones`, accessed on 8/11/2019

[31] `http://www.unicode.org/cldr/charts/25/supplemental/languages_and_scripts.html`, accessed on 7/11/2019

**Language population, mln (4 bins):**
#1: *less than 1*, 265
#2: *between 1 and 10*, 183
#3: *between 10 and 100*, 66
#4: *100 and more*, 13
(not specified), 87

**Plural rules (15 cat, multiple):**
#1: *cardinal;one,other*, 117
#2: *ordinal;other*, 49
#3: *cardinal;other*, 26
#4: *ordinal;one,other*, 10
#5: *cardinal;one,few,many,other*, 8
...
(none of them), 440

**Number of characters (5 bins):**
#1: *less than 26*, 34
#2: *between 26 and 29*, 35
#3: *between 30 and 49*, 79
#4: *between 50 and 100*, 38
#5: *100 and more*, 6
(not specified), 422

Figure 2.6: Handcrafted descriptors for languages in the Unicodelang data set (the number of instances in each category of each descriptor as well as the amount of unclassified instances, where it is applicable, are shown in grey color)

types (cardinal, ordinal *etc.*) and plural rules' categories (one, many, other *etc.*), were extracted from the languages' supplementary information[32]. The first attribute, plural rules' types, is used unchanged, while the second, plural rules' categories, was aggregated with the first to create a new feature, "plural rules" (see Fig. 2.6 for an example). The last property values, number of characters in the alphabet, were parsed from the additional supplementary materials for languages[33,34] and placed into five categories with meaningful ranges (Fig. 2.6).

### Network construction

As before, we used the Tanimoto coefficient to build similarity layers, and we received acceptable similarity values (last two rows in Table 2.3). Similarly to MovieLens, we introduced a threshold to binarise bipartite interactions. If the percentage of population speaking given language is more or equal to 0.01% than the example is considered to be positive or negative otherwise, giving us 819 positive and 436 negative edges in the network. As an result, we obtain a three layer graph with the network characteristics as in Table 2.4 (see Fig. 2.7 for an illustration). As in the case of the MoveLens network, the resulting Unicodelang graph has an acceptable sparsity, and it is fully connected.

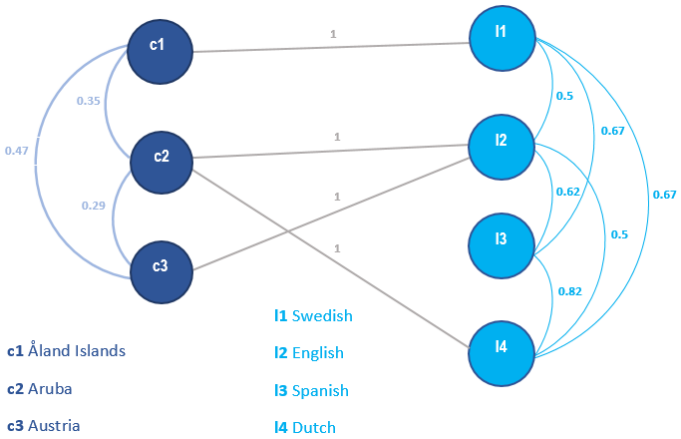


Figure 2.7: An extract from the Unicodelang network with three layers represented by different colors: the bipartite layer is in grey (Unicode languages), country similarity network is in smoked blue and language similarity network is in light blue

[32] `http://www.unicode.org/cldr/charts/25/supplemental/language_plural_rules.html`, accessed on 7/11/2019

[33] Using Main Letters/Native attribute

[34] `http://www.unicode.org/cldr/charts/25/summary`, accessed on 7/11/2019

# 3. Validation of link prediction founded on external knowledge

*This chapter describes a framework for validating drug-target predictions based on an external resource. I present the principle first, which includes a description of the external resource, the idea, and measures used in the validation framework. I then present the framework that consists of the algorithm, result normalization process and is followed by an example. I finish the chapter with the conclusion. The framework is our main contribution in this chapter.*

## 3.1 Principle

In this section, I describe the Unified Medical Language System (UMLS) [21], real label knowledge base regularly curated by the National Institutes of Health (NIH)[1], which is used for validation. I then explain the motivation and the idea of validation of link prediction based on the external resource. I finish by presenting the semantic relatedness measures, the essential part of the validation process.

### 3.1.1 The UMLS

In this work, we use the UMLS as the most reliable resource that can be found in an open access. Curated twice a year by NIH, it has no alternative among professionally curated knowledge bases on medical domain. The other popular resources based on ontologies, i.e. BioPortal [121] or The Open Biological and Biomedical Ontology (OBO) Foundry [146], are curated by users that makes the solutions developed with the use of such resources less reliable. We use the UMLS knowledge to evaluate relationship between drug-target pairs, predicted by our approaches discussed in other chapters to be interacting.

The UMLS is a repository of several biomedical vocabularies from NIH and other sources. Vocabularies are selected lists of words and phrases used for tagging units of information to make them easily retrieved by a search. The modern version of the UMLS integrates more than 14 million biomedical names for about 4 million concepts from over 150 families of vocabularies, and more than 12 million relations among these concepts. The system is composed of three knowledge sources: the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon (Fig. 3.1).

[1] `https://www.nih.gov`

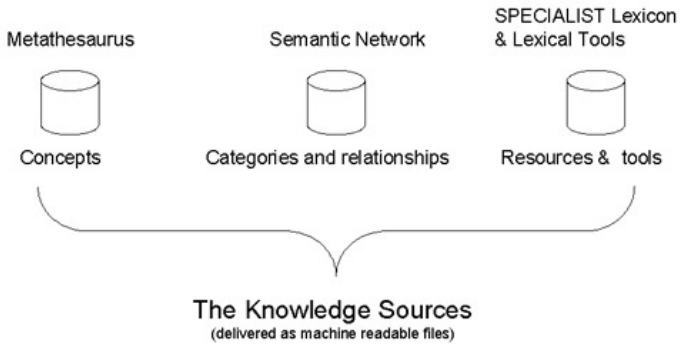


Figure 3.1: The structure of the UMLS [21]

In addition to data, the UMLS includes tools for querying the Metathesaurus (MetamorphoSys) and for extracting UMLS concepts from text (MetaMap [48]).

The Metathesaurus is a vocabulary database containing information about biomedical and healthcare concepts including their names, aliases and relationships. The database is composed of many types of biomedical vocabularies such as The Systematized Nomenclature of Medicine (SNOMED)[2], Online Mendelian Inheritance in Man (OMIM)[3], the Medical Subject Headings (MeSH)[4], The National Cancer Institute Thesaurus (NCI Thesaurus)[5] and others (Fig. 3.2). A fundamental unit in the Metathesaurus is *concept*, which represents a single meaning and contains all atoms from any source that express that meaning in any format (see Fig. 3.3 for an example).

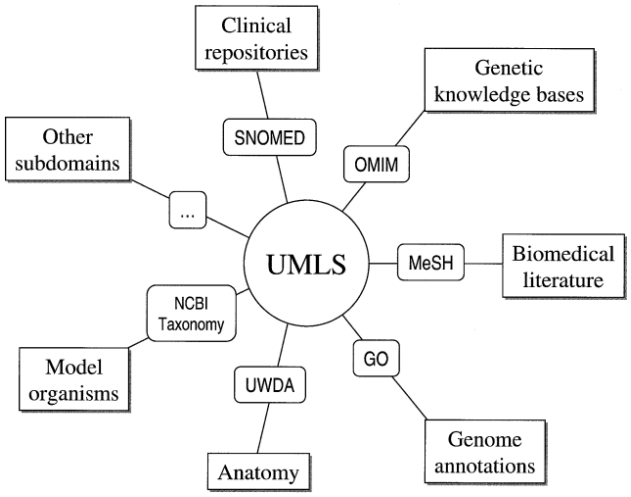


Figure 3.2: The various vocabularies and domains integrated in the Metathesaurus [21]

The Metathesaurus data is collected from different sources including classifications, codes, terms used in public health, biomedical literature and research information. Names stored in the database are organized into concepts by meaning, each of which is assigned a unique identifier and placed in the database structure. This structure has several levels of specification: CUI (Concept Unique Identifiers), AUI (Atom Unique Identifiers) *etc.* Atoms are the basic building blocks from which the Metathesaurus is constructed. They represent concepts taken from a particular vocabulary

[2] `http://www.snomed.org`
[3] `http://www.ncbi.nlm.nih.gov/omim`
[4] `https://www.nlm.nih.gov/mesh`
[5] `http://ncit.nci.nih.gov`

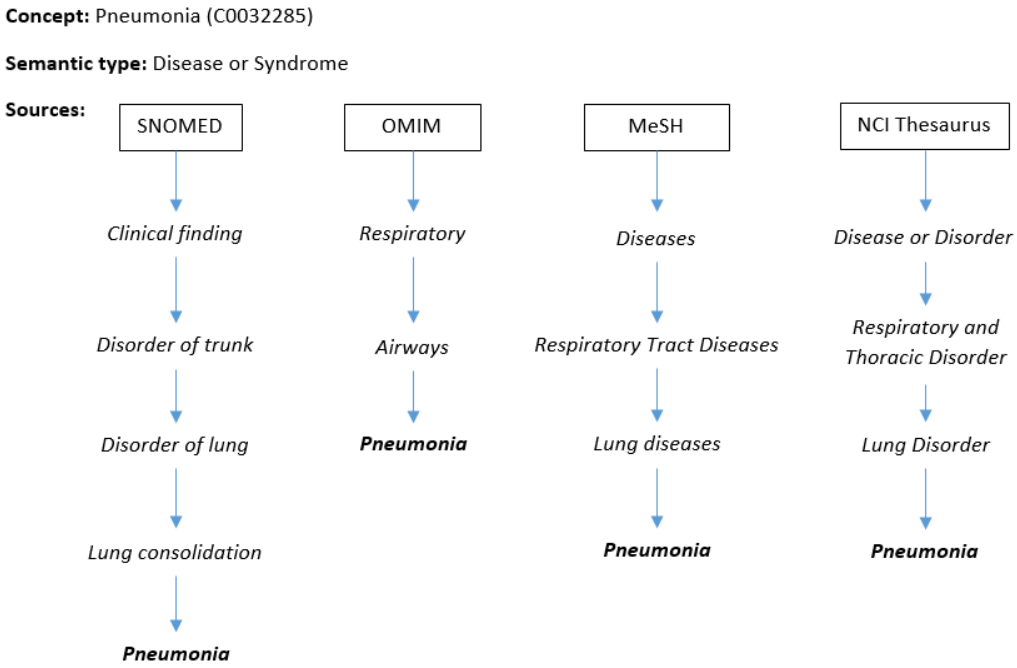


Figure 3.3: A simplified example of a concept and its basic properties: semantic type, sources and ancestors

source. To be more precise, every occurrence of a concept in each source vocabulary is considered to be an atom and assigned a unique identifier or "AUI". We do not focus on other levels of specification since we only use identifiers of type "AUI" in this work (see Section 3.2.1).

The Semantic Network provides categorization of concepts presented in the Metathesaurus [22]. The network consists of semantic types (high level categories like "Disease or Syndrome" or "Clinical Drug") and semantic relationships (relationships between semantic types, for example: "Clinical Drug" treats "Disease or Syndrome"). Every Metathesaurus concept is assigned at least one semantic type. Semantic types and semantic relationships create a network that represents the biomedical domain (Fig. 3.4). The information associated with each semantic type includes: unique

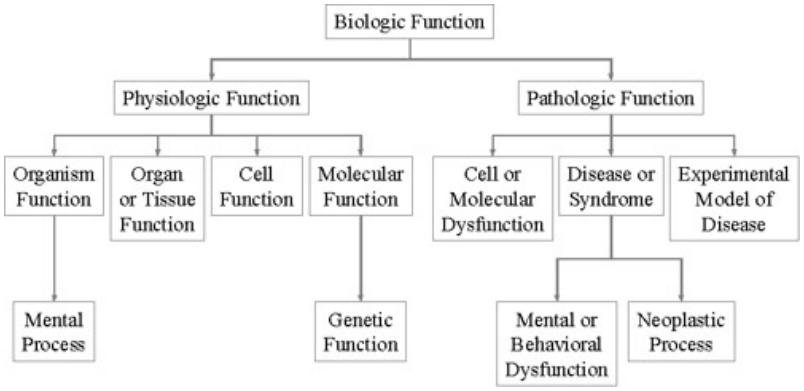


Figure 3.4: Some semantic types of the Semantic Network [21]

identifier, definition, its parent and children. The information associated with each relationship includes: unique identifier, semantic type, definition, examples *etc.* The network has over fifty semantic relationships, the most common of which is the isa relationship. The *isa* relationship creates the hierarchy of types within the Semantic Network and is used for assigning the most

specific type to a concept in the Metathesaurus. Some examples of the "isa" relationship are:

- "Animal" *isa* "Entity"
- "Carbohydrate" *isa* "Chemical"
- "Human" *isa* "Mammal"

Semantic relationships can go in agreement with the concept level or not. For example, the relationship "Clinical Drug" *causes* "Disease or Syndrome" does not hold at the concept level for "Aspirin" and "Cancer". Aspirin does not cause cancer. Simultaneously, not all relationships that apply at the concept level are present in the Semantic Network.

In this chapter, we exploit the Metathesaurus and the Semantic Network to evaluate relationship between given drug-target pair. We match drugs and targets with the Metathesaurus concepts, examine their relationships in the Semantic Network, and give feedback depending on how they are positioned in the hierarchy (not connected at all, directly connected, not directly *etc.*). In the next section, we develop this idea by providing more details on the evaluation framework.

### 3.1.2 The validation idea

The main target of external validation is the evaluation of the quality of drug-target prediction performed on multi-layer networks by using an external resource such as the UMLS. The key idea of this validation is to use a resource with concepts associated to drugs and targets in order to express the strength of the relationship between a given drug-target pair, and then to compare the results with ones of a given method. The estimation of the strength of predicted drugs and targets is performed by computing the similarity of associated concepts, and the quality of prediction is expressed by a confidence score.

The ultimate goal of using an evaluation framework is to evaluate a given interaction prediction method (such as the ones developed in Chapter 5 and 8). The method outputs drug-target pairs which are predicted to be interacting, and the framework labels each of these pairs with a score. In order to compare the results provided by a prediction method and the framework, score values need to be binarised. To satisfy the latter, we introduce a *threshold*: if the strength of the interaction is more or equal to the threshold value, we consider an interaction as verified, and not otherwise. By using a single threshold, each measure becomes a binary event: "an interaction is verified" or "it is not". We treat each measure as a probability score and to keep a balance between two classes we use 0.5 as a threshold value. Taking into account that confidence scores we use are normalized from 0 to 1, it allows us to interpret any interaction which strength satisfies the selected threshold as not worse than random guessing.

We use this framework to evaluate our methods presented later in Chapter 5 and 8. As an evaluation criterion we report the percentage of verified predictions passing a certain threshold.

### 3.1.3 Assessing the strength of interactions

In order to assess the strength of the relationship between a given drug and a given target, we first match both the drug and the target to the concepts from the given external resource, then to compute the strength we use a semantic relatedness measure [60]. As such measure we use one of the existing semantic similarity measures. In addition to them, we propose our own metric, the Non-zero score.

#### Existing semantic measures

We implement and test the three most common semantic similarity measures from the literature: Path similarity [136], the Lin similarity [103] and the Semantic vector similarity [112].

**Path similarity** *Path similarity* is defined inversely proportional to the distance between concepts in the hierarchical representation:

$$sim_{path}(c_1, c_2) = \frac{1}{p}, \tag{3.1}$$

where $p$ is the number of nodes in the shortest path between concepts $c_1$ and $c_2$ inclusive the nodes.

According to this measure, the further two concepts are positioned towards each other in the hierarchy, the less similar are the concepts.

*Example* Consider the concept hierarchy as in Fig. 3.5. Path similarity between two concepts *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)* is equal to $1/9$, because there are 9 nodes in the shortest path between the concepts (shown as $N_1, N_2, \ldots, N_9$ in Fig. 3.5):

$$sim_{path}(\text{C0558141}, \text{C3811388}) = \frac{1}{9}. \tag{3.2}$$

Note that we do not count semantic types in the shortest path, only concepts associated with the types (for more details please see Section 3.2.3).

**Lin similarity** The *Lin similarity* is based on a notion of *information content* (IC). IC of a concept quantifies the information the concept contains [140]:

$$IC(c) = -\log\left(\frac{\frac{|leaves(c)|}{|subsumers(c)|}+1}{max\_leaves+1}\right), \tag{3.3}$$

where $|leaves(c)|$ denotes the number of *leaves* (child concepts without following children) related to concept $c$, and $|subsumers(c)|$ is the number of *subsumers* (concept ancestors) related to concept $c$. In other words, the more leaves a concept has relative to the number of ancestors, the less information is included in the concept. This is normalized by *max_leaves*, the total number of leaves in the hierarchy. Note, by the number of subsumers of a concept we understand the concept itself and the number of all its ancestors.

Using a notion of IC, the Lin similarity can be defined as:

$$sim_{lin}(c_1, c_2) = \frac{2 \cdot IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)}, \tag{3.4}$$

where $lcs(c_1, c_2)$ is *least common subsumer* (LCS) of $c_1$ and $c_2$. The Lin measure compares IC of a pair of concepts to IC of their least common subsumer. The higher the IC value, the more specific the subsumer is, and the more similar the concepts are.

Note that for leaves IC will take the form of Eq. 3.5, a constant value depending on the number of leaves in the hierarchy:

$$IC(c) = -\log\left(\frac{1}{max\_leaves+1}\right), \tag{3.5}$$

Taking into account that concepts associated with drugs and targets are leaves (they do not have following children), and that IC of a leaf is a constant, the Lin similarity almost becomes the Resnik similarity [140]:

$$sim_{res}(c_1, c_2) = IC(lcs(c_1, c_2)). \tag{3.6}$$

The only difference between the Lin similarity and the Resnik similarity is that Lin is normalized by logarithm of a constant, producing values in the range from 0 to 1 and allowing to use it as a probability function.

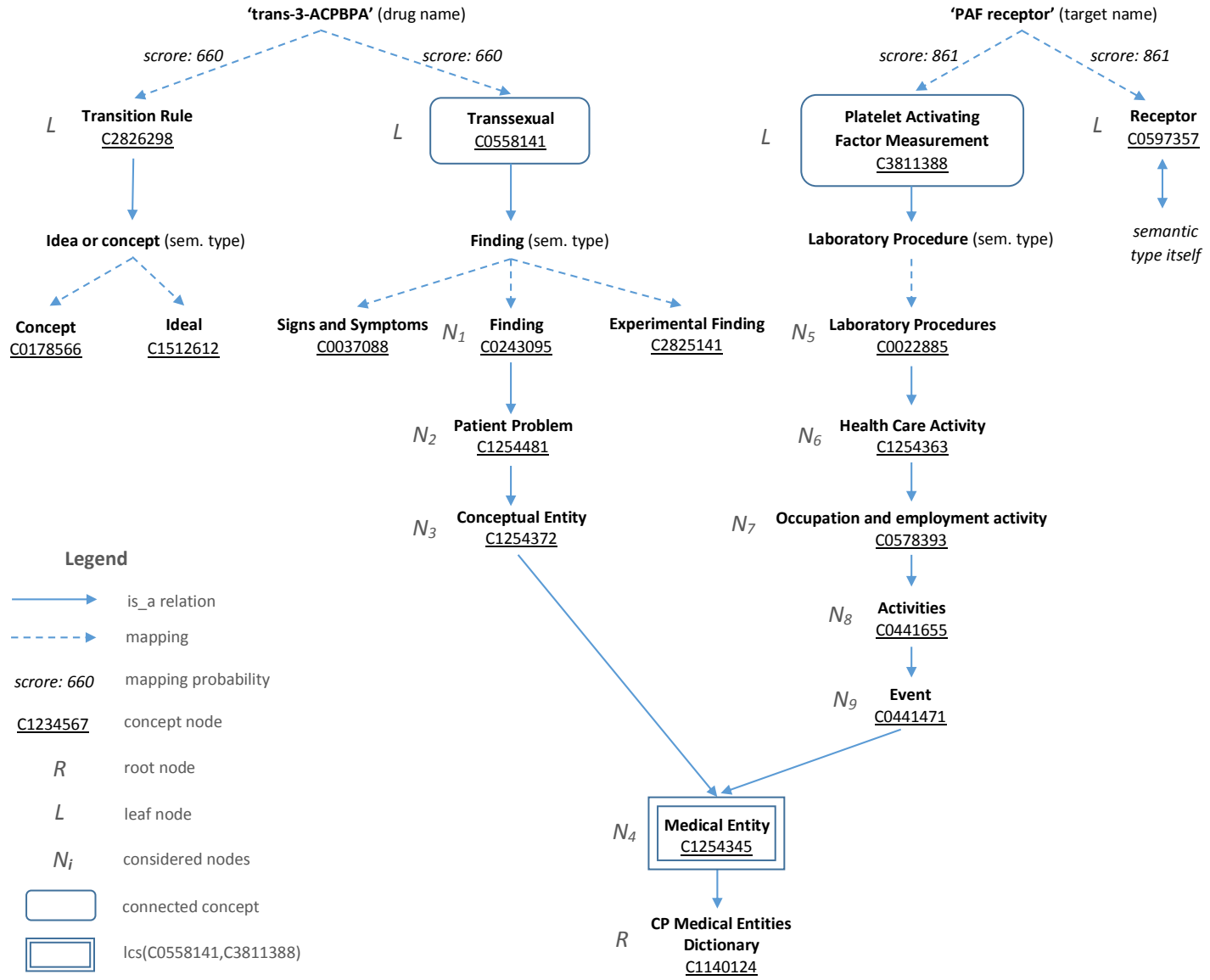


Figure 3.5: UMLS concepts hierarchy representation for an example drug-target pair: 'trans-3-ACPBPA' used as a drug and 'PAF receptor' as a target

*Example* Given a concept hierarchy as in Fig. 3.5, the Lin similarity between concepts *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)* can be computed as following. First, we compute the information content of the concepts' LCS, which is *Medical Entity (C1254345)* in this example (shown as *lcs(C0558141,C3811388)* in Fig. 3.5). We assume that LCS has 9506 leaves (all leaves in Fig. 3.5 are shown as L), and the maximum number of leaves in the hierarchy is 9508. Using Eq. 3.3 IC of LCS can be computed as:

$$IC(\text{C1254345}) = -\log\left(\frac{\frac{9506}{2}+1}{9508+1}\right) = 0.301,$$

where 2 is the number of subsumers of LCS (node R and LCS itself in Fig. 3.5). Once IC of LCS is obtained, one needs to compute IC of the two concepts. As it was noted above, information content of a leaf node (such as *C0558141*, *C3811388*) is equal to a constant depending on maximum number of leaves in the hierarchy:

$$IC(\text{C0558141}) = IC(\text{C3811388}) = -\log\left(\frac{1}{9508+1}\right) = 3.978.$$

Now semantic similarity of *C0558141* and *C3811388* can be computed using Eq. 3.4:

$$sim_{lin}(\text{C0558141},\text{C3811388}) = \frac{2\cdot 0.301}{3.978+3.978} = 0.076. \tag{3.7}$$

**Semantic vector similarity** In the *Semantic vector similarity* two concepts must be represented by vectors using the same principle as in Information retrieval. According to this principle, one concept is considered to be a document and another a query, and the concept similarity is computed in the same way as document similarity. To construct vector representation of the two concepts, every node contained in either concept's hierarchy receives a value of $1/(1+d)$ in the vector coordinate related to this node, where $d$ is the shortest distance between the node and the second concept. After this step, the semantic vector similarity is defined as the cosine between two semantic vectors representing a pair of concepts:

$$sim_{vec}(c_1,c_2) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}, \tag{3.8}$$

where $A_i$ and $B_i$ are semantic vectors of concepts $c_1$ and $c_2$ respectively, and $n$ equals the number of all concepts involved in both vectors.

In such a way, the vectors occurring from any two concepts will have a non-orthogonal component proportional to the depth of their least common ancestor, and the cosine function will indicate the closeness of the semantic content for the two compared vectors.

*Example* As in previous examples, let us compute the Semantic vector similarity between concepts *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)* on the concept hierarchy as in Fig. 3.5.

First, one needs to construct vector representation of concepts *C0558141* and *C3811388*. The process is similar to one used in document retrieval. To create this representation we build shortest distance matrix of the two concepts (Table 3.1), which helps us to encode the vectors.

Once the shortest distance matrix is built one can compute vector representation of the concepts. Table 3.2 shows this representation. Each cell in the table corresponds to a vector coordinate computed as $1/(1+d)$, where $d$ is the shortest distance between concepts, and the number of coordinates corresponds to the number of nodes in the hierarchy related to the two concepts.

Table 3.1: The shortest distance matrix for concepts C0558141 ($C_1$) and C3811388 ($C_2$) with $R$ as a root node and $N_i$ considered concept nodes (Fig. 3.5)

| | $C_1$ | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $R$ | $C_2$ | $N_5$ | $N_6$ | $N_7$ | $N_8$ | $N_9$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_1$ | 0 | 1 | 2 | 3 | 4 | 5 | – | – | – | – | – | – |
| $C_2$ | – | – | – | – | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 |

Table 3.2: Vector representation of concepts C0558141 ($C_1$) and C3811388 ($C_2$) with $R$ as a root node and $N_i$ considered concept nodes (Fig. 3.5)

| | $C_1$ | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $R$ | $C_2$ | $N_5$ | $N_6$ | $N_7$ | $N_8$ | $N_9$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_1$ | $\frac{1}{1+0}$ | $\frac{1}{1+1}$ | $\frac{1}{1+2}$ | $\frac{1}{1+3}$ | $\frac{1}{1+4}$ | $\frac{1}{1+5}$ | – | – | – | – | – | – |
| $C_2$ | – | – | – | – | $\frac{1}{1+6}$ | $\frac{1}{1+7}$ | $\frac{1}{1+0}$ | $\frac{1}{1+1}$ | $\frac{1}{1+2}$ | $\frac{1}{1+3}$ | $\frac{1}{1+4}$ | $\frac{1}{1+5}$ |

Now using Table 3.2, where row $C_1$ represents $A_i$ in Eq. 3.8 and $C_2$ same for $B_i$ respectively, semantic similarity between *C0558141* and *C3811388* can be computed as:

$$sim_{vec}(\text{C0558141}, \text{C3811388}) = \\ = \frac{(\frac{1}{5}\cdot\frac{1}{7})+(\frac{1}{6}\cdot\frac{1}{8})}{\sqrt{1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+\frac{1}{36}}\cdot\sqrt{1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+\frac{1}{36}+\frac{1}{49}+\frac{1}{64}}} = 0.033, \tag{3.9}$$

taking into account that "–" in Table 3.2 corresponds to 0.

#### Additional baseline measure

In addition to measures existing in the literature, we add our own measure, the Non-zero score, which is very simple. It is simpler than any of the measures presented above, what allows us to use it as a baseline. We assess behavior of our method also using this baseline measure.

**Non-zero score** We define the *Non-zero score* as an indication function showing if two concepts have a common subsumer in the concepts hierarchy. The function returns 1 when the common subsumer exists and 0 otherwise:

$$score_{nzero}(c_1, c_2) = \begin{cases} 0, & \nexists\, lcs(c_1, c_2) \\ 1, & \exists\, lcs(c_1, c_2) \end{cases} . \tag{3.10}$$

We mainly use this measure to report the percentage of concept pairs which can be theoretically validated with a semantic relatedness measure on a given database of concepts.

*Example* For the concept hierarchy as in Fig. 3.5, the Non-zero score of concepts *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)* is 1, because the concepts have a common subsumer, *Medical Entity (C1254345)* in this particular example:

$$score_{nzero}(\text{C0558141}, \text{C3811388}) = 1.$$

Contrariwise, the Non-zero score of concepts *Transition Rule (C2826298)* and *Receptor (C0597357)* for the same hierarchy is 0, because they do not have LCS:

$$score_{nzero}(\text{C2826298}, \text{C0597357}) = 0.$$

Note that from an implementation point of view it is not necessary to compute this score explicitly. Its value can be obtained as a side effect from using any other measure (Path, Lin or Semantic vectors): if a concept pair has non-zero value for any similarity measure the Non-zero

score will be 1, and 0 otherwise. By definition, the other measures we use are dependent on the presence of LCS: in the absence of that their values automatically become 0, and if LCS exists, they cannot be 0 at all. Such relation might be considered as weak, that is why we interpret the Non-zero score as a baseline measure.

## 3.2 The framework to validate link prediction

In this section, I present the framework that I use for validating predicted interactions using the UMLS as an external knowledge resource. It includes the description of the algorithm and the result normalization. I then provide an example of how the algorithm and similarity measures are computed in practice.

### 3.2.1 The algorithm

We define the verification as a multi-step process described below:

1. Match drugs and targets from predicted interactions with the Metathesaurus vocabulary. For that, we convert the *IUPHAR drug/target id* into the *IUPHAR drug/target provisional name* (see Section 2.1.1);
2. Since there is no direct interactions between items of the Metathesaurus vocabulary, we introduce a level of abstraction by retrieving semantic types for all predicted drugs and targets. For this step, we use the MetaMap tool[6];
3. Retrieve semantic type relations of type *inverse_isa* by querying the Metathesaurus in a recursive way. Filter results by type of concept names "AUI" (to exclude distinct and normalized concept names) and "ENG" language (to avoid redundancy when using multiple sources);
4. For each predicted drug-target pair compute a confidence score defined as:

$$score_{sim}(d,t) = \max_{i \in S, j \in T} \left( \frac{conf(d_i)}{1000} \cdot \frac{conf(t_j)}{1000} \cdot sim(d_i,t_j) \right), \tag{3.11}$$

   where $conf(d_i)$, $conf(t_j)$ denote MetaMap confidence scores of a drug concept $i$ and a target concept $j$, $sim(d_i,t_j)$ is one of the semantic similarity measures presented in 3.1.3, and $S$, $T$ are the numbers of retrieved semantic types for drug and target respectively (see Section 3.2.3 for examples);
5. Generate feedback for each predicted drug-target pair (see Section 3.2.3 for an example), and compute precision for all predictions, which is defined as a ratio between the number of verified interaction pairs (i.e. passed selected threshold) among predicted pairs to the total number of predicted pairs:

$$Prec\ (sim) = \frac{TP\ (sim)}{TP\ (sim) + FP\ (sim)}, \tag{3.12}$$

   where $TP\ (sim)$ represents the number of interactions verified by the UMLS w.r.t. measure of type *sim*, and $FP\ (sim)$ is the number of interactions which did not. Please note, taking into account that UMLS is not complete (not all possible biomedical concepts are presented and not all even presented concepts are fully detailed, i.e. concept ancestors might not be available *etc.*), the value of $TP\ (sim)$ is under estimated. Precision computed in Eq. 3.12 can only increase if the UMLS was complete. We discuss on this point in the next section.

Note that steps 1-3 can be precomputed for all drugs and targets in a data set. The simplified version of the algorithm for a given drug-target pair is presented in Algorithm 1.

[6] `https://metamap.nlm.nih.gov`

**Algorithm 1:** The UMLS validation algorithm

**Input** : drug $d$, target $t$, similarity measure *sim*
**Output** : $(d,t)$ is verified (boolean)
$c_d \leftarrow \text{MetaMap}(d)$
$c_t \leftarrow \text{MetaMap}(t)$
**for** $d_i$ **in** $c_d$ **do**
  **for** $t_j$ **in** $c_t$ **do**
    $score \leftarrow (conf(d_i)/1000) \cdot (conf(t_j)/1000) \cdot sim(d_i,t_j)$
  **end**
**end**
**if** $max(score) < 0.5$ **then**
  **return** $False$
**else**
  **return** $True$
**end**

### 3.2.2 Result normalization

In this section, we propose to additionally normalize precision in order to make an adjustment for the limits of the UMLS caused by the fact that the knowledge base is not complete. In other words, not every (even existing) drug-target pair can be verified within the system, and the normalization that we propose in Eq. 3.13 can exclude these cases:

$$Prec\ (sim)^{norm} = \frac{Prec\ (sim)}{h_{max}(sim)}, \tag{3.13}$$

where $Prec\ (sim)$ represents precision related to measure *sim*, and $h_{max}$ is maximum response of IUPHAR on the UMLS w.r.t. measure *sim*. We use the value of the latter to quantify the limit of the UMLS.

We explore the limit of the UMLS by performing the evaluation using the algorithm described in Section 3.2.1 as before, but instead of predicted interactions we use *all* possible combinations of drug-target pairs from IUPHAR. By doing that, we would like to estimate which percentage of links unknown by IUPHAR *cannot be* verified by the UMLS (because of the absence of the LCS between two given concepts in the hierarchy due to the incompleteness of the UMLS). To quantify this percentage numerically, we define *maximum response* as a ratio of the number of verified interactions to the total number of examined interactions:

$$h_{max}(sim) = \frac{I_{all\text{-}ext}\ (sim)^{verif}}{I_{all} - I_{ext}}. \tag{3.14}$$

$I_{all\text{-}ext}\ (sim)^{verif}$ in numerator of Eq. 3.14 represents the number of interactions verified by the framework using *sim* measure from the set of all possible combinations of drug-target pairs in IUPHAR (*all*) minus known existing interactions (*ext*). $I_{all}$ in the denominator defines the total number of interactions possible with IUPHAR which can be computed as: $I_{all} = |V_d| \cdot |V_t|$, and $I_{ext}$ is the number of already known interactions in IUPHAR: $I_{ext} = |\{(u,v)|u \in V_d, v \in V_t, (u,v) \in E_{V_d V_t}\}|$.

Maximum response ($h_{max}$) represents optimal prediction that can be achieved by an ideal method (i.e. having accuracy of prediction 100%). Its value depends on the two parameters: database used (IUPHAR *etc.*) and semantic similarity measure. Note that, since we test our framework only with IUPHAR, the denominator of Eq. 3.14 takes a constant value that is equal to: $8978 \cdot 2910 - 12456 = 26113524$ (see Section 6.4.1 for more details).

**Example of normalization** To give an example of the normalization process, imagine a network consisting of 10 drugs, 10 targets and 10 interactions between them. The total number of possible interactions in this network will be equal to: $I_{all} = |V_d| \cdot |V_t| = 10 \cdot 10 = 100$, the number of existing interactions: $I_{ext} = 10$, and the total number of considered interactions: $I_{all} - I_{ext} = 100 - 10 = 90$. Also imagine that the UMLS with given similarity measure can only verify 45 out of 90 considered interactions: $I_{all\text{-}ext}^{\ verif} = 45$. Now, maximum response can be obtained using Eq. 3.14:

$$h_{max} = \frac{45}{100 - 10} = 50\%. \tag{3.15}$$

Imagine also that our method (for instance NEWERMINE) predicts 30 new not existing interactions, but only 12 of them can be verified by the UMLS. This means that $TP = 12$ (interactions verified by the UMLS), $FP = 18$ (interactions not verified by the UMLS) and $TP + FP = 30$. Using Eq. 3.12, precision of our method can be computed as:

$$Prec = \frac{12}{12 + 18} = 40\%. \tag{3.16}$$

Using maximum response computed in Eq. 3.15 normalized precision can be obtained as:

$$Prec^{\ norm} = \frac{0.4}{0.5} = 80\%. \tag{3.17}$$

This means that the prediction method has 80% of precision taking into account the limitations of the UMLS.

In this thesis, we compute precision with each measure defined in Section 3.1.3. We report both normalized and non-normalized values. The results are summarized in the experiments for our methods in Chapter 6 and 9.

### 3.2.3 Validation example

Let us consider an example to illustrate how our framework works and confidence scores based on different measures are computed in practice. Given *"trans-3-ACPBPA"* and *"PAF receptor"* as a drug and a target provisional names respectively, we would like to compute the strength of their relationship using the algorithm described in Section 3.2.1 with semantic relatedness measures presented in Section 3.1.3.

**Concept hierarchy** Fig. 3.5 illustrates the concept hierarchy for the given drug-target pair. The hierarchy is shown upside down in the way it was build: the validating drug and target are on the top and the root node on the bottom. According to the output of MetaMap *"trans-3-ACPBPA"* is matched with concepts *Transition Rule (C2826298)* and *Transsexual (C0558141)* with score 660 for each matching, and *"PAF receptor"* to concepts *Platelet Activating Factor Measurement (C3811388)* and *Receptor (C0597357)* with score 861 for each. Matching score represents similarity of matched concept with the querying name: the matching is exact if score is 1000, and not exact if score is less than 1000. In our example the matching is not exact for both names (i.e. with the scores 660 and 861 for the first and the second respectively), and we take this information into account to compute the confidence score based on Path, Lin and Semantic vector similarity measures.

**Non-zero score** Exploring the hierarchy further we can find out that *"trans-3-ACPBPA"* and *"PAF receptor"* have only one common node in the concepts hierarchy of UMLS – *Medical Entity (C1254345)*, which is *LCS* for the two concepts, *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)*. It automatically means that the Non-zero score (defined in Eq. 3.10) of validating drug-target pair is 1, because the two concepts the pair is associated with have a common subsumer:

$$score_{nzero}(\text{"trans-3-ACPBPA"}, \text{"PAF receptor"}) = 1.$$

**Path similarity** Confidence score based on Path similarity depends on the matching scores of drug and target names to the associated concepts and the similarity between the concepts. Path similarity for concepts *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)* associated with *"trans-3-ACPBPA"* and *"PAF receptor"* was already computed in Eq. 3.2. Now we can compute confidence score for our drug-target pair using Eq. 3.11:

$$score_{path}(\text{"trans-3-ACPBPA"}, \text{"PAF receptor"}) = \frac{660}{1000} \cdot \frac{861}{1000} \cdot \frac{1}{9} = 0.063.$$

Note that we do not take into account semantic type when computing the shortest path between the concepts (Fig. 3.5). We do not consider it because semantic type itself is not present in the hierarchy of concepts of the Metathesaurus but in the Semantic Network, and we use MetaMap to retrieve concepts associated with the type.

**Lin similarity** As in case with Path similarity, the confidence score based on Lin also depends on the matching scores of drug and target names to the concepts and the similarity between the two. The latter was already computed in Eq. 3.7 for *Transsexual (C0558141)* and *Platelet Activating Factor Measurement (C3811388)*, the concepts associated with the validating pair. Analogously to Path, the confidence score based on the Lin similarity can be computed using Eq. 3.11:

$$score_{lin}(\text{"trans-3-ACPBPA"}, \text{"PAF receptor"}) = \frac{660}{1000} \cdot \frac{861}{1000} \cdot 0.076 = 0.043.$$

**Semantic vector similarity** As in previous examples the confidence score based on the Semantic vector similarity can be computed using Eq. 3.11 and the precomputed value of similarity measure for the associated concepts *C0558141* and *C3811388* from Eq. 3.9:

$$score_{vec}(\text{"trans-3-ACPBPA"}, \text{"PAF receptor"}) = \frac{660}{1000} \cdot \frac{861}{1000} \cdot 0.033 = 0.019.$$

**Conclusion** To sum up, by providing this example we demonstrated how to perform validation on the UMLS of a link predicted from IUPHAR by using our algorithm. We performed that with three existing similarity measures (Path similarity, the Lin similarity, the Semantic vector similarity) and got three different confidence scores based on that, and in addition we used our own Non-zero score as a baseline. Using values we received and a thresholding function defined in Section 3.1.2, our interaction can be considered as verified with the Non-zero score only. The result provided by the Non-zero score is more optimistic than the confidence scores based on other measures, because their values are less than 0.5 (1, 0.063, 0.043, 0.019 for Non-zero, Path, Lin, the Semantic vector score respectively). The results provided by other measures cannot pass the selected threshold, because the two concepts associated with a drug-target pair are not exact (i.e. matching with a score $< 1000$), and there is a long path between these concepts making their common subsumer too general.

## 3.3 Conclusion

We developed the framework for validation of predicted drug-target interactions founded on an external resource. We adopted our framework to the UMLS, which we use as a knowledge base. Other knowledge sources could be used instead of the UMLS, for instance BioPortal or OBO Foundry, but they are not curated by an academic institution, and thus they are less reliable.

The proposed example demonstrated that our framework can be used to perform validation of predicted interactions. Detailed results with the use of the framework on the real data are presented and discussed later in Chapters 6 and 9.

# II Predicting links with random walks


**4 Classical random walker** . . . . . 45
4.1 Related work
4.2 Random walks on multi-layer network
4.3 Conclusion

**5 PageRank models** . . . . . 51
5.1 Related work
5.2 The NEWERMINE algorithm
5.3 Conclusion

**6 Experimental evaluation of random walk methods** . . . . . 63
6.1 Experimental settings
6.2 Experimental results
6.3 Sparsification experiment
6.4 Validation of link prediction with external knowledge
6.5 Conclusion

**7 Analysis of connectivity characteristics** . . . . . 71
7.1 Motivation
7.2 Experimental settings
7.3 Experimental evaluation
7.4 Conclusion

# 4. Classical random walker

*In this chapter, I address the link prediction problem defined in Chapter 1 focusing on classical random walker approaches without matrix formulations. First, I present the state of the art of this group of methods. I then describe a general approach for performing link prediction by randomly walking in a single graph. Next, I extend this approach to the case of multi-layer graphs, our problem setting defined in Chapter 1. This includes the description of the algorithm and is followed by a computational example. I then finish the chapter with the conclusion. The resulting algorithm, which we call* EXPLICIT RANDOM WALKER*, is the main contribution in this chapter. I use this algorithm as a baseline in Chapter 6 to compare with a more advanced approach developed in Chapter 5.*

## 4.1 Related work

In this section, I first present the literature overview on random walks in general and in relation to the link prediction task. I then describe the random walk model, an approach for link prediction in single graphs.

### 4.1.1 State of the art

The simulation of a random walk is a fundamental problem in algorithmic theory, which has many applications in various fields from searching for an exit in a maze [15] to analyzing a biological network [118] and protocol testing [29]. The classical theory of random walks deals with random walks on simple infinite graphs (having infinite number of vertices) such as grids. In this work, we limit ourselves to the use of a random walker in graphs with finite number of vertices with a focus on the link prediction task.

The random walk is a well-known method for graph exploration [2, 4, 41, 62]. First introduced by Pearson *et al.* in [130], it had been widely studied by a number of scientists during the past century. Lovasz *et al.* in [109] surveyed different aspects of the theory of random walks on graphs, which included qualitative behavior, bounds of important parameters related to time, relationships with matrix eigenvalues and electrical networks and some other aspects. According to the authors of the survey, the random walker simulates a dust particle that randomly moves in the room having finite set of states. The classical theory of random walks studies probabilities of such a walker to return to its starting state or to arrive to a given state before it visits all the states. The probabilities

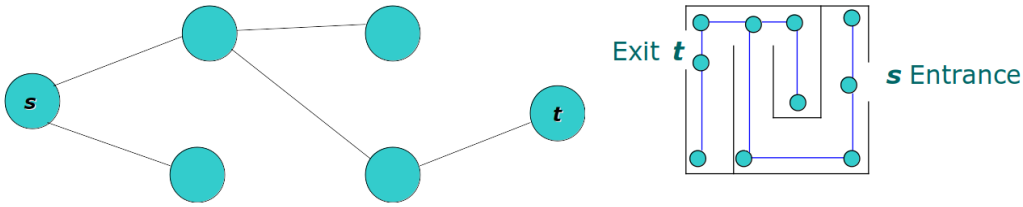


Figure 4.1: An example of the s-t connectivity problem (left) and its application in searching a path from the entrance to the exit in a maze (right) [139]

of such events are far from zero if certain conditions are satisfied [109]. The studied properties allowed to use this model as the link prediction framework in graphs.

Link prediction via random walk in graphs takes its origin from solving the *s-t connectivity* problem. The objective of this problem is to decide whether two given vertices $s$ and $t$ of a graph are in the same connected component of this graph (Fig. 4.1). Karlin *et al.* in [88] survey random walk solutions to this problem. The main idea of these algorithms is that if target node $t$ is reached from $s$ by doing random walk in a limited number of steps, then the querying vertices are in the same connected component. Can *et al.* in [29] reformulate the problem of link prediction as proximity computation for the two query nodes. According to the authors, the proximity can be estimated by a percentage of time spent for traversing graph by doing random walk from the first query node to the second, with time defined as number of iterations. In this approach, the higher the proximity between two query nodes, the higher the probability of a link between them.

Classical random walker solutions deal with single graphs (i.e. having edges of one type only), while our problem setting defined in Section 1.2.2 requires multi-layer graph input. In Section 4.2.1, we present an algorithm which provides a solution to this, but before that we would like to provide an explanation on how link prediction via random walk in graphs works in more detail.

### 4.1.2 The random walk model for link prediction

In this section, we focus on the use of random walk for link prediction. The ideas for solving the s-t connectivity problem by random walking from [88] and proximity estimation problem from [29] can help to derive a general approach for link prediction in a single graph.

The model, which we refer to as *the random walk model*, can be defined as follows. Starting from a randomly selected node, the walker performs moves along edges of the graph at random. At each step, the edge to follow is chosen uniformly from all outgoing links of the current node (in the case of an unweighted graph) or proportional to link weights (in the case of a weighted graph). How frequently the walker visits the node affects node importance: a node with higher frequency is considered more important than a node with a low value (Fig. 4.2).

In order to improve network exploration, this strategy can be modified in a number of ways: the walker can be constrained to perform at most $k$ steps, to not visit any of the last $r$ vertices it encountered, or with small probability $1-\beta$ the process can be restarted at any time to avoid getting trapped by isolated vertices or vertices without outgoing edges (in the case of a directed graph). The product of the probabilities of edges the walker traversed gives the cumulative probability of a path between two nodes and can be used to predict the link between a starting node and an end node: if the path probability is greater than a given threshold, a new edge is predicted. The process must be repeated until the desired target vertex is reached or the maximum number of steps have been performed. If not limiting the number of steps, the process might last too long due to its randomized nature and as a result the cumulative probability of such prediction will be very low regardless the real existence of the link. To avoid this, random walks are usually repeated several times to derive more robust estimates.

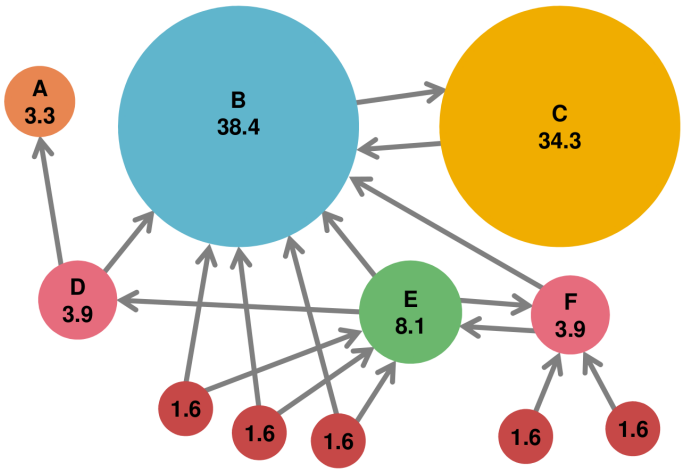


Figure 4.2: Nodes importance example in a graph, taken from [98]

The random walk model allows to solve the link prediction problem for single graphs. The extension of that to multi-layer graphs without the use of matrix formulations is still possible by using a naive approach only. In the next section, we propose an algorithm developing such an approach.

## 4.2 Random walks on multi-layer network

In this section, I extend the random walk model from Section 4.1.2 to the multi-layer graph setting described in Section 1.2.2. I present the resulting algorithm first, then provide an example.

### 4.2.1 Explicit Random Walker

The random walk model described in Section 4.1.2 cannot be used directly with our problem setting defined in Chapter 1 and needs to be adapted accordingly. To solve this, we propose a framework for multi-layer graphs that we call EXPLICIT RANDOM WALKER. According to our proposal, every time the walker selects an edge where to go next it needs to choose a layer first, and only then selects an edge to follow from the chosen layer. The layer is selected uniformly at random from the set of layers the current node is present in. In that case the cumulative path probability must be updated accordingly: current values are multiplied by $\frac{1}{|G_l^m|}$, where the denominator represents the number of layers in the multi-layer graph in which the current node is present out of $|\{G_d^i\}|+|\{G_t^j\}|+|\{G_{dt}^k\}|$ possible (see Def. 1.1.12 for more details). The way how the layer is selected is the main difference between our framework and the random walk model from the previous section.

Trying to develop this idea with respect to the task of drug-target interaction prediction, we derived the following set of important steps, which will be later summarized in the formal algorithm:

1. Select drug as a starting node $d_i$. Initialize *path_strength* by 1. Initialize the *current_node* by $d_i$.
2. Decide with probability $\beta$ to continue the walk or with $1-\beta$ to return to $d_i$. Probability to restart the walk $1-\beta$ must be small, for example in the interval $[0.05, 0.3]$. To satisfy that value of $\beta$ should be in the interval $[0.7, 0.95]$.
3. If one decides to continue the walk, randomly select a network from the set of networks the *current_node* is connected in, update *network* variable. Also update the *path_strength* by multiplying its current value by the probability of the *network* taking into account that every

**Algorithm 2:** The EXPLICIT RANDOM WALKER algorithm

**Input :** multi-layer graph $G$, starting node $d_i$, target node $t_j$, maximum number of steps $k$, probability to restart $\beta$, number of last visited nodes $r$

**Output :** $path\_strength$ as probability of $(d_i, t_j) \in G$

$path\_strength \leftarrow 1$
$current\_node \leftarrow d_i$
$visited\_nodes \leftarrow \emptyset$
$step \leftarrow 0$
**while** $(step < k)$ *and* $(current\_node \neq t_j)$ **do**
  **if** $random() < \beta$ **then**
    $network \leftarrow G_p$ **with** $current\_node \in V_{G_p}, p = random()$
    $path\_strength \leftarrow path\_strength \cdot \beta \cdot \frac{1}{|G_l^m|}$ **with** $current\_node \in V_{G_l^m}$
    $next\_node \leftarrow v$ **with** $v = random(), (current\_node, v) \in E_{G_{network}}$
    **if** $next\_node$ **not in** $visited\_nodes$ **then**
      $current\_node \leftarrow next\_node$
      $path\_strength \leftarrow path\_strength \cdot probability(next\_node)$
      $visited\_nodes \leftarrow visited\_nodes \cup current\_node$
      **if** $length(visited\_nodes) > r$ **then**
        $visited\_nodes \leftarrow visited\_nodes[1 : r+1]$
      **end**
    **end**
  **else**
    $current\_node \leftarrow d_i$
    $path\_strength \leftarrow path\_strength \cdot (1 - \beta)$
  **end**
  $step \leftarrow step + 1$
**end**
**return** $path\_strength$

network has an equal chance to be selected (for example, if *current_node* is connected in 3 networks, every of these networks has probability of $^1/_3$ to be chosen).

4. Compute edge probabilities for the *network* taking into account that every edge has an equal probability to be chosen. For weighted edges, edge probabilities must be multiplied by edge weights and normalized to sum to 1 (for example, if *current_node* has 4 edges in the *network* with weights 0.2, 0.8, 0.8, 0.2 for each edge respectively, every of these edges has probability $^1/_4$ to be chosen, and their weighted probabilities, after the normalization process, will become 0.1, 0.4, 0.4, 0.1 summing to 1, for each edge respectively).
5. Select an edge and the corresponding *next_node* at random taking into account the edge probabilities computed at the previous step. Perform the move to the *next_node* by updating the *current_node*. Update the *path_probability* by multiplying its current value by weighted edge probability computed before.
6. Repeat steps 2-5 until the given *target_node* is reached or the maximum number of steps is used.

The walk, performed by steps defined above, must be repeated enough number of times (for instance, 1000 or more) reporting the maximum *path_strength* obtained among all walks at the end. We use maximum as an average function to get the most optimistic result, which, analogously

with the shortest part, will correspond to the most probable path between the two vertices. Another improvement can be made to speed up the process. At every iteration one can maintain the list of $r$ last visited nodes that will be used as follows. If *next_node* is presented in this list, the iteration must be skipped. This improvement helps to avoid local minima that can lead to visit the same nodes multiple times.

The EXPLICIT RANDOM WALKER algorithm is summarized in Algorithm 2. The presented algorithm is a simple extension that is straightforward to implement. However, this extension results in a high computational cost (to predict one link the walk must be repeated *explicitly* a number of times, which is a high number in practice, and it results in a high complexity). We will use this algorithm as a baseline approach later in Chapter 6.

### 4.2.2 Example

Let us consider a simplified example on how EXPLICIT RANDOM WALKER works in practice. Consider a bipartite multi-layer network as shown in Fig. 1.3. We would like to predict if an interaction between drug $l1$ and target $t4$ is possible in 5 steps at most, and what its probability is? We assume that the threshold for predicting a link as positive is 0.0001.

The network presented in Fig. 1.3 has 5 drug nodes: $l1$, $l2$, $l3$, $l4$, $l5$, 4 target nodes: $t1$, $t2$, $t3$, $t4$ and edges from 6 different layers: IUPHAR, DrugBank, BioGrid, drug-similarity, target-similarity (substrings) and target-similarity (motifs). The network is weighted with edge weights as shown in the figure.

We run Algorithm 2 on our example. For simplicity reasons, we do not take into account probability for restart $\beta$ and limit our walk to 5 steps maximum and maintain a list with 5 visited nodes. Due to the random processes involved in the algorithm, its iterations are not deterministic. The possible output might be, for example, the following:

1. Initialization:
   starting node $d_i$: l1
   target node $t_j$: t4
   *path_strength* = 1
   *visited_nodes* = {l1}

2. Iteration 1:
   *step* = 0
   *current_node*: l1
   *network*: drug-similarity
   *next_node*: l3, not in *visited_nodes*, continue
   $path\_strength = path\_strength \cdot network\_probability \cdot probability(next\_node)$
   $path\_strength = 1 \cdot 1 \cdot \frac{0.66}{0.66+0.2+0.42+0.43} = 0.39$
   *visited_nodes* = {l1, l3}

3. Iteration 2:
   *step* = 1
   *current_node*: l3
   *selected_network*: drug-similarity
   *next_node*: l2, not in *visited_nodes*, continue
   $path\_strength = 0.39 \cdot 1 \cdot \frac{0.23}{0.23+0.32+0.65+0.66} = 0.047$
   *visited_nodes* = {l1, l3, l2}

4. Iteration 3:
   *step* = 2
   *current_node*: l2
   *selected_network*: IUPHAR
   *next_node*: t2, not in *visited_nodes*, continue
   *path_strength* = $0.047 \cdot \frac{1}{2} \cdot 1 = 0.0235$
   *visited_nodes* = {l1, l3, l2, t2}

5. Iteration 4:
   *step* = 3
   *current_node*: t2
   *selected_network*: target-similarity (motifs)
   *next_node*: t3, not in *visited_nodes*, continue
   *path_strength* = $0.0235 \cdot \frac{1}{4} \cdot \frac{0.75}{0.75+0.87+0.71} = 0.0019$
   *visited_nodes* = {l1, l3, l2, t2, t3}

6. Iteration 5:
   *step* = 4
   *current_node*: t3
   *selected_network*: target-similarity (substrings)
   *next_node*: t4, not in *visited_nodes*, continue
   *path_strength* = $0.0019 \cdot \frac{1}{3} \cdot \frac{0.64}{0.64+0.7+0.69} = 0.0002$
   *visited_nodes* = {l3, l2, t2, t3, t4}

The target node is reached in 5 steps, the algorithm stops. The result is positive: the interaction between $l1$ and $t4$ is possible with path probability 0.0002. The link class is predicted as True, because the path probability is more than a given threshold.

To produce more reliable estimates, the iterations in the algorithm must be repeated many times for each predicted pair taking a maximum of the path probability values at the end. The resulting values then can be used to rank the predicted pairs as well as to predict the exact classes of the links, depending on the settings of the task.

## 4.3 Conclusion

The proposed example shows that the EXPLICIT RANDOM WALKER algorithm is very strait to implement, which can be considered as its advantage. Its another advantage is in that the algorithm has possibility of result interpretability, which can be implemented, for instance, by logging the walk as in the example in Section 4.2.2 and backtracking the nodes and their connections after making prediction. At the same time, the fact that each iteration must be performed explicitly and repeated many times introduces additional expenses into the algorithm's computational cost. In the next chapter, we will develop an approach based on matrix multiplications, which is not dependent on this limitation and as a result more efficient. We therefore keep EXPLICIT RANDOM WALKER as a baseline to make a comparison of these two approaches in Chapter 6.

# 5. PageRank models

*In this chapter, I address the link prediction problem defined in Chapter 1 focusing on random walker approaches with matrix formulations, usually referred as PageRank models. First, I present the state of the art of this group of methods. I then present the PageRank algorithm in general terms and explain how it can be used to predict links in single-layer graphs. Next, I describe the state of the art approach* NRWRH*, an extension of PageRank for multi-layer graphs with 3 layers. Finally, I present our proposition,* NEWERMINE*, as an extension of PageRank for multi-layer graphs with any number of layers, which is the case of our problem setting defined in Chapter 1. This includes the description of the algorithm and an example. I finish the chapter with the conclusion. The* NEWERMINE *algorithm is the main contribution in this chapter, also published at IDA'18.*

## 5.1 Related work

In this section, I first present the state of the art for link prediction in graphs focusing on PageRank based approaches. I then describe the original PageRank algorithm, list its extensions that we will use later and explain how it can be transferred to the link prediction task in single graphs. A the end of the section, I present the state of the art approach NRWRH extending the PageRank idea to the multi-layer graph setting with 3 layers.

### 5.1.1 State of the art

As discussed in Section 4.1.1, the random walk is a long-established method for network exploration. The classical approaches, i.e. without matrix formulations, are easy to understand and implement. This also means that they can easily be extended to some specific cases, e.g. to multi-layer networks, as we showed in Section 4.2 of the previous chapter. The main drawback of the classical methods is high running time. As noted at the end of the example in Section 4.2.2, the iterations must be repeated many times. It requires a lot of running time and results in low efficiency and increased computational cost. The matrix based methods, which we refer to as the PageRank group of methods, resolve this problem.

A significant advance in the simulation of random walks was made by Page and Brin, the founders of Google, in their scientific report [128], where the details of the PageRank algorithm were described. The use of matrix computations brought the simulation of a random walker to a new level. The modeling of a random walker going to all directions at the same time significantly

improved efficiency and opened many prospects for the following modifications. The most notable among them are Topic-sensitive PageRank [74], Personalized PageRank [85], Weighted PageRank [169] and Fast Random Walk with Restart [158], allowed more control of the random walker behavior by biasing the walk (the first two modifications) or by exploiting characteristics of the network (the last two).

Originally developed for vertex ranking task, PageRank can be easily adapted to perform link prediction in graphs. Random walker models based on matrix computations are widely used to deal with this problem. The main application domains include social networks [7, 179], recommender systems [176], protein activity prediction [57], drug-target interaction prediction [106, 111] and many others [104].

Recent PageRank adaptations expand link prediction task to the multi-layer bipartite graphs, the main objective of this thesis. These types of methods are dependent on fixing the number of networks, usually limited to three: two similarity networks of both types of vertices and a bipartite layer. In case of more layers some of the networks are proposed to be aggregated to fit the limitation requirements. A notable work on this direction is Chen *et al.* [34]. Dealing with drug-target interaction prediction, the authors propose to calculate the drug-similarity network as weighted average of two similarity measures. They combine the resulting network with the target similarity layer and the interaction network into a three-layer network, which they refer to as "heterogeneous". The random walk with restart is simulated by matrix multiplication in a fashion of PageRank. The authors show that the results are getting worse if one ignores the interaction network or when a single similarity layer is used. Their algorithm called NRWRH is presented and discussed further in Section 5.1.4.

Another example with a limited number of layers is by Xie *et al.* [167]. The authors use a three layer network in gene-disease prediction. They simulate random walk by matrix multiplication, using different numbers of steps for the two similarity networks. Using a similar bi-random walk idea, Luo *et al.* in [111] predict microRNA-disease interactions, exploiting a three-layer network. Liu *et al.* in [107] work on the same task. They use symmetric random walk with restart, functionally the same as in [34], with the similarity networks constructed by averaging two similarity measures.

The main challenge of current PageRank adaptations for the multi-layer bipartite graph setting remains the same – how to choose the best aggregation mechanism to fit the required number of layers.

### 5.1.2 PageRank algorithm

As was said before in Section 5.1.1, random walk can be simulated via matrix multiplication, significantly improving its running time. Due to the nature of the matrices used, computations in such approaches are performed in parallel, which significantly speeds up the process. In most of the matrix adaptations, it is realized with the PageRank algorithm.

#### Classical algorithm

The classical PageRank models the behavior of a random walker in matrix form. Originally, it was used to estimate importance of web pages on the Internet, where the Internet was modeled as a graph every vertex of which represents a web page, and edges represent links between pages (see Fig. 1.4 for a simplified example of a graph representing the Internet). According to the PageRank idea, the random walker is assumed to be a surfer who browses pages by randomly selecting the links from one page to another. The pages visited more often are considered more important than rarely visited ones, and consequently they are ranked higher by the algorithm (Fig. 4.2). This information was used by a search engine such as Google to decide which pages to output first for a given user search query.

Going into the details of PageRank, the Internet graph is modeled by a transition matrix $M$ defined as described in Def. 1.1.15, and a random surfer is initialized by the initial state vector. The transition matrix describes what happens with a surfer on the next step. More precisely, each element $M_{ij}$ of this matrix represents the probability of a random surfer to come from page $i$ to page $j$.

The initial state is defined by vector $\mathbf{v}_0$, which is $n \times 1$ dimensional. The elements $v_i$ of $\mathbf{v}_0$ have equal values $1/n$, which represents the idea that a random surfer can start its browsing from every page with equal probability $1/n$ (an example of the initial state vector $\mathbf{v}_0$ for the example graph from Fig. 1.4 is shown in Eq. 5.1).

$$\mathbf{v}_0 = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \tag{5.1}$$

The next state vector $\mathbf{v}_{k+1}$ is obtained by multiplying matrix $M$ with the previous state vector $\mathbf{v}_k$. Every value $v_i$ of the new state vector gives the probability of a random surfer starting from the initial state to arrive at page $i$ on the iteration $\mathbf{v}_{k+1}$. An important feature of PageRank is that by multiplying matrix $M$ by current state vector $\mathbf{v}$ enough number of times, convergence will be reached[1] in a way that the current state vector $\mathbf{v}$ is not changing more than a given tolerance threshold (see Eq. 5.2 for the example graph from Fig. 1.4).

$$\begin{aligned} \mathbf{v}_1 = M \cdot \mathbf{v}_0 = \begin{bmatrix} 1/3 \\ 1/6 \\ 1/6 \\ 1/3 \end{bmatrix}, \mathbf{v}_2 = M \cdot (M \cdot \mathbf{v}_0) = \begin{bmatrix} 15/54 \\ 2/9 \\ 2/9 \\ 15/54 \end{bmatrix}, \\ \mathbf{v}_3 = M \cdot (M \cdot (M \cdot \mathbf{v}_0)) = \begin{bmatrix} 51/162 \\ 30/162 \\ 30/162 \\ 51/162 \end{bmatrix}, \dots, \\ \mathbf{v}_n = M \cdot \dots (M \cdot (M \cdot (M \cdot \mathbf{v}_0)) \dots) = \begin{bmatrix} 3/10 \\ 1/5 \\ 1/5 \\ 3/10 \end{bmatrix}. \end{aligned} \tag{5.2}$$

When convergence is reached, the resulting vector $\mathbf{v}$, representing a final state of the "infinite" walk, gives probabilities for all pages of a random surfer to finish its walk starting from a random page. The pages with higher values are considered to be more important by the algorithm and will be ranked higher by the search engine. Notably, the vector values sum up to 1, this is why the value $v_i$ gives natural probability of a random surfer to arrive on page $i$ at the end.

Another important property of PageRank is that in practice the number of iterations is quite small even for graphs with huge number of vertices making the algorithm very efficient for a given task [98].

[1] if certain conditions are satisfied: graph is undirected, in case of directed graph it must be *irreducible*, meaning that it has a path from every of its nodes to every other node, and *aperiodic*, i.e. the greatest common divisor of the lengths of its cycles is one [143]

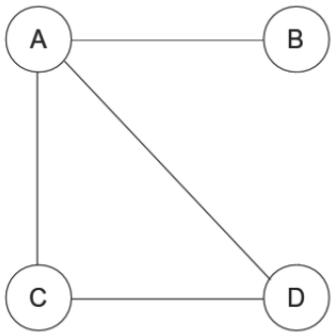


Figure 5.1: Example graph to illustrate link prediction with PageRank

#### PageRank modifications

Analogously to the random walk model from Section 4.2, PageRank can be modified to be able to avoid *dead ends* (vertices with no edge out) and *spider traps* (sets of nodes with no dead ends but no links out). For the first, probability to restart $\beta$ can be used in the same way as before, for the latter, *taxation* can be implemented by adding to the calculation a small probability 1-$\beta$ of the random surfer teleporting to a random vertex (Eq. 5.3).

$$\mathbf{v}_{k+1} = \beta\, M \cdot \mathbf{v}_k + (1-\beta)\, \mathbf{v}_0 \tag{5.3}$$

This idea can be extended to a *Personalized* PageRank, where vector $\mathbf{v}_0$ in Eq. 5.3 is replaced by another one with not equal values, which makes the random walk process biased [75].

Some other modifications on PageRank are possible, but we are not focusing on them since they are not used in this work.

### 5.1.3 PageRank for link prediction

PageRank can be used as a solution to link prediction task in graphs. If one initializes the initial state vector $\mathbf{v}_0$ with 1 in $v_i$, corresponding to vertex $i$, and zeros elsewhere, the resulting vector $\mathbf{v}$, when convergence is reached, will give probabilities for every page of a random surfer to finish its walk in it starting from page $i$. This idea can be exploited for estimating the link probability of a none-existing edge in a graph.

To illustrate how this method works, consider a graph from Fig. 1.4 with the removed edge between vertices B and D as shown in Fig. 5.1.

Let us try to estimate probability of the link between edges B and D. According to Def. 1.1.15, the transition matrix $M$ for the given graph and the initial state vector $\mathbf{v}_0$ will be as in Eq. 5.4, and the iteration process will take a form as in Eq. 5.5.

$$M = \begin{bmatrix} 0 & 1 & 1/2 & 1/3 \\ 1/3 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 1/3 \\ 1/3 & 0 & 1/2 & 0 \end{bmatrix}, \mathbf{v}_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{5.4}$$

$$\mathbf{v}_1 = M \cdot \mathbf{v}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v}_2 = M \cdot (M \cdot \mathbf{v}_0) = \begin{bmatrix} 0 \\ 1/3 \\ 1/3 \\ 1/3 \end{bmatrix},$$
$$\mathbf{v}_3 = M \cdot (M \cdot (M \cdot \mathbf{v}_0)) = \begin{bmatrix} 11/18 \\ 1/9 \\ 1/9 \\ 1/9 \end{bmatrix}, \ldots, \tag{5.5}$$
$$\mathbf{v}_n = M \cdot \ldots (M \cdot (M \cdot (M \cdot \mathbf{v}_0)) \ldots) = \begin{bmatrix} 3/8 \\ 1/5 \\ 1/5 \\ 9/40 \end{bmatrix}.$$

According to the final state vector in Eq. 5.5, the probability of a link between B and D will be estimated as 9/40. Taking into account that there are only 2 new links possible from vertex B: to vertex C and to vertex D, the latter option is the most probable one in this example (probability of the first is 1/5 and $9/40 > 1/5$).

To conclude, the given example demonstrates how the PageRank algorithm can be used to predict links in a single graph. In the next section, we will discuss an extension of this idea to a special case of a multi-layer graph.

### 5.1.4 The NRWRH algorithm

The idea of using PageRank to solve the link prediction task in a single-layer graph, where only one edge is possible between any two vertices, cannot be used directly with our problem setting defined in Section 1.2.2 and needs to be adapted to the case of a multi-layer network. The state of the art algorithm NRWRH described below partially solves this problem.

Chen *et al.* in [34] proposed a solution to the link prediction problem in a bipartite multi-layer network with 3 layers: drug similarity, target similarity and drug-target interaction networks. They refer to this setting as *heterogeneous* network, since all these networks have different or *heterogeneous* origin and thus should be represented separately. In their algorithm, which they refer to as NRWRH (Network-based Random Walker with Restart on Heterogeneous network), the transition matrix $M$ and state vector $\mathbf{v}$ are defined in the manner described in 5.1.2, but taking into account heterogeneous layers:

$$M = \begin{bmatrix} M_{tt} & M_{td} \\ M_{dt} & M_{dd} \end{bmatrix}, \mathbf{v} = \begin{bmatrix} (1-\eta)\,\mathbf{v}_t \\ \eta\,\mathbf{v}_d \end{bmatrix}, \tag{5.6}$$

where $M_{dd}$ – transition matrix for drug similarity layer, $M_{tt}$ – transition matrix for target similarity layer and $M_{dt}$, $M_{td}$ – transition matrices for drug-target and target-drug interaction layers, $\mathbf{v}_d$, $\mathbf{v}_t$ – state vectors for drugs and targets respectively, and $\eta \in [0,1]$ – parameter weighting the importance of drug network and target network. The whole matrix $M$ has dimensionality $m \times n$, where $n$ – number of drugs and $m$ – number of targets. The method also has an additional user-specified parameter $\lambda \in [0,1]$ affecting moves in the bipartite layer from drugs to targets and vice versa.

The algorithm iterates as follows. For a given drug $d_i$ it performs a search of best target candidates among not directly connected target vertices to $d_i$. For that $\mathbf{v}_{d_0}$ is initialized by assigning 1 at the position for a given drug $d_i$ and 0 elsewhere, and $\mathbf{v}_{t_0}$ is initialized by assigning $\frac{1}{|\{(v,u)\in E_{dt}\}|}$ at the positions for the targets linked to drug $d_i$ and 0 otherwise. The iteration process looks the same as in classical PageRank with *taxation* and probability to restart $\beta$ (Eq. 5.3). The iterations are

running until $||\mathbf{v}_{k+1} - \mathbf{v}_k||_1 < 10^{-10}$, where $||_1$ operation denotes $L_1$-norm. Targets with maximum value in $\mathbf{v}_t$ among all those not directly connected to $d_i$ are considered as more probable targets for the drug $d_i$.

## 5.2 The NEWERMINE algorithm

In this section, I present the NEWERMINE algorithm, an extension of PageRank for multi-layer graphs with any number of layers that I have developed. It is free from the limitations which other methods mentioned in Section 5.1.1 share and satisfies our problem setting defined in Chapter 1. I first describe the algorithm, and the construction of matrices and vectors used by it. Then I provide an example on how the algorithm is computed in practice.

### 5.2.1 The algorithm

The major drawback of the NRWRH algorithm presented in Section 5.1.4 is that it is limited to 3 layers, and it is not clear how to overcome the case when the number of layers is different from that. To solve this problem, we adapt this approach to a setting with $|\{G_d^i\}| + |\{G_t^j\}| + |\{G_{dt}^k\}| \geq 3$. The algorithm stays essentially the same. The main changes affect construction of the transition matrix, state vector and removal of the user-dependent parameter $\lambda$.

We decompose the transition matrix $M$ from Section 5.1.2 into matrix $M_0$ for within-network/layer transitions, and introduce another matrix $N$ for between-network/layer transitions. Explicitly creating $M$ in the way described in 5.1.4 is easy for the 3 layer setting, but becomes much more complicated with different number of layers. We therefore construct $M_0$ by positioning $M_{G_t}$, $M_{G_d}$ and $M_{G_{td}}$ on the main diagonal of a new matrix with zeros elsewhere as shown in Eq. 5.7:

$$M_0 = \begin{bmatrix} M_{G_t} & 0 & 0 \\ 0 & M_{G_d} & 0 \\ 0 & 0 & M_{G_{td}} \end{bmatrix}, \tag{5.7}$$

where $M_{G_t}, M_{G_d}, M_{G_{td}}$ are derived from target similarity, drug similarity and drug-target interaction networks respectively (Eq. 5.8-5.10).

$$M_{G_t} = \begin{bmatrix} M_{G_t}^1 & 0 & \dots & 0 \\ 0 & M_{G_t}^2 & \dots & 0 \\ 0 & 0 & \dots & M_{G_t}^{|\{G_t^j\}|} \end{bmatrix} \tag{5.8}$$

$$M_{G_d} = \begin{bmatrix} M_{G_d}^1 & 0 & \dots & 0 \\ 0 & M_{G_d}^2 & \dots & 0 \\ 0 & 0 & \dots & M_{G_d}^{|\{G_d^i\}|} \end{bmatrix} \tag{5.9}$$

$$M_{G_{td}} = \begin{bmatrix} M_{G_{td}}^1 & 0 & \dots & 0 \\ 0 & M_{G_{td}}^2 & \dots & 0 \\ 0 & 0 & \dots & M_{G_{td}}^{|\{G_{td}^k\}|} \end{bmatrix} \tag{5.10}$$

The transition matrix $N$, in its turn, explicitly models possible layer transitions, with 1 on the main diagonal of a submatrix $N_{G_j \to G_i}$ for all nodes present in both layers, and 0 otherwise (Eq. 5.11).

**Algorithm 3:** The NEWERMINE algorithm

**Input :** transition matrix for within-layer transitions $M_0$, transition matrix for between-layer transitions $N$, starting node $d_i$, maximum number of steps $k$, network importance $\eta$, probability to restart $\beta$, threshold $\delta$

**Output :** probability scores $\mathbf{v}$

$\mathbf{v}_{0_d} \leftarrow$ initialize with $d_i$

$\mathbf{v}_{0_t} \leftarrow$ initialize targets for which an interaction with $d_i$ is known

$\mathbf{v}_0 \leftarrow (1-\eta)\ {\mathbf{v}_{0_d}}^{norm} + \eta\ {\mathbf{v}_{0_t}}^{norm}$

$step \leftarrow 0$

**repeat**

  $step \leftarrow step + 1$

  $\mathbf{v}_{step} \leftarrow \beta\ (M_0 \cdot N)^{norm} \cdot \mathbf{v}_{step-1} + (1-\beta)\ \mathbf{v}_0$

**until** $(\sum |\boldsymbol{v}_{step} - \boldsymbol{v}_{step-1}| \leqslant \delta) \vee (step > k)$

**return** $\mathbf{v}_{step}$

$$N = \begin{bmatrix} N_{G_t^1 \to G_t^1} & N_{G_t^2 \to G_t^1} & \cdots & N_{G_{td}^{|\{G_{td}^k\}|} \to G_t^1} \\ \cdots & \cdots & \cdots & \cdots \\ N_{G_t^1 \to G_{td}^{|\{G_{td}^k\}|}} & N_{G_t^2 \to G_{td}^{|\{G_{td}^k\}|}} & \cdots & N_{G_{td}^{|\{G_{td}^k\}|} \to G_{td}^{|\{G_{td}^k\}|}} \end{bmatrix} \tag{5.11}$$

This also means that transition matrices from drug to target layers (and vice versa) have zeros everywhere including the main diagonal.

Finally, we construct the state vector with entries for *all* vertices in *all* layers as shown in Eq. 5.12:

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_{G_t} \\ \mathbf{v}_{G_d} \\ \mathbf{v}_{G_{td}} \end{bmatrix}, \tag{5.12}$$

where $\mathbf{v}_{G_t}$, $\mathbf{v}_{G_d}$, $\mathbf{v}_{G_{td}}$ – state vectors of target, drug and drug-target interaction networks accordingly (Eq. 5.13).

$$\mathbf{v}_{G_t} = \begin{bmatrix} {\mathbf{v}_{G_t}}^1 \\ {\mathbf{v}_{G_t}}^2 \\ \cdots \\ \mathbf{v}_{G_t^{|\{G_t^j\}|}} \end{bmatrix}, \mathbf{v}_{G_d} = \begin{bmatrix} {\mathbf{v}_{G_d}}^1 \\ {\mathbf{v}_{G_d}}^2 \\ \cdots \\ \mathbf{v}_{G_d^{|\{G_d^i\}|}} \end{bmatrix}, \mathbf{v}_{G_{td}} = \begin{bmatrix} {\mathbf{v}_{G_{td}}}^1 \\ {\mathbf{v}_{G_{td}}}^2 \\ \cdots \\ \mathbf{v}_{G_{td}^{|\{G_{td}^k\}|}} \end{bmatrix} \tag{5.13}$$

The initial state vector $\mathbf{v}$ has dimensionality $|V_t||\{G_t^j\}| + |V_d||\{G_d^i\}| + |V_d \cup V_t||\{G_{dt}^k\}|$, which is equal to a one dimension of transition matrices $M$ and $N$. As in NRWRH, we initialize the initial state vector $\mathbf{v}_0$ by setting the entry for the starting drug and each linked target to 1 in every network they are present. Note, matrices and state vectors are column-normalized – the entries of a column must sum to 1.

The algorithm, that we called **NE**t**W**ork-bas**E**d **R**andom walk on **M**ult**I**-layered **NE**twork or NEWERMINE, is summarized in Algorithm 3. The iteration process of NEWERMINE is similar to the one in NRWRH – the algorithm stops when the sum of values in the difference between the current state vector and the previous one becomes less than a certain predefined threshold: $|\mathbf{v}_{k+1} - \mathbf{v}_k| \leqslant \delta$. At the end of computations, the final state vector $\mathbf{v}$ needs to be summarized by

summing up for each vertex all corresponding entries, leading to a vector with dimensionality $|V_d \cup V_t|$ from which the edge ranking can be derived.

The product $(M_0 \cdot N)^{norm}$ from Algorithm 3 can be precomputed, giving a matrix that is functionally equivalent to $M$ in NRWRH. More precisely, the new precomputed matrix models the behavior of a random walker selecting a network where to go uniformly at random from the list of networks available for the given vertex, and deciding which vertex edge to follow then, also at random taking weights of edges into account. Doing that precomputation in practice also saves computation time for every iteration.

### 5.2.2 Computational example

Let us consider a simplified example to illustrate how matrices $M$ and $N$ of NEWERMINE are constructed and iteration process works step by step. Given an unweighted multi-layered network with two target, one drug and one drug-target interaction networks shown as in Fig. 5.2, we would like to verify if an interaction between drug $l2$ and target $t4$ is possible, assuming the threshold 0.1 for predicting a link as positive and the iteration process threshold $10^{-3}$. Note, for simplicity reasons the example graph is unweighted (all edges are labeled by 1), while in this work we work with weighted networks. In addition, we ignore the effects of parameters $\beta$ and $\eta$ (in other words, parameters are fixed to: $\beta = 1$ and $\eta = 0$).

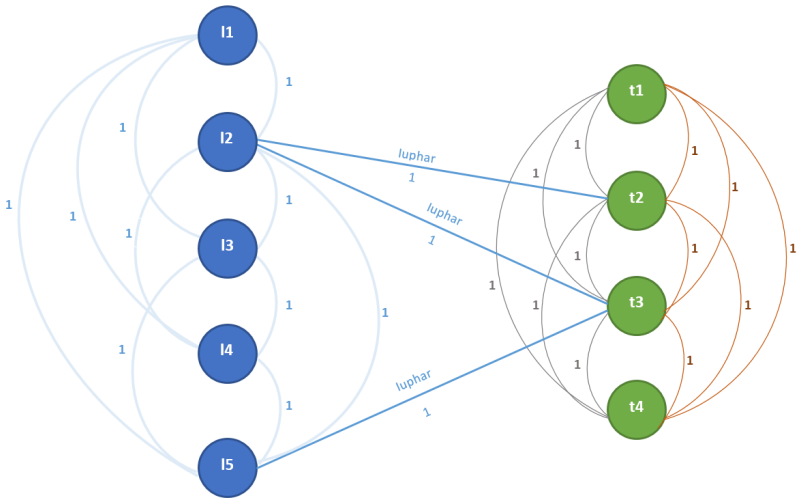


Figure 5.2: An example of a multi-layer graph with 4 layers: drug interaction network is in light blue, two target interaction networks are in brown and grey, and one bipartite network is in deep blue

The transition matrix for within-layer transitions of the given graph will take the form as in Eq. 5.14, in which different colors represent different layers in accordance with Fig. 5.2. There are four "boxes" on the main diagonal of this matrix, each of which represents different layers: first target similarity, second target similarity, drug similarity and drug-target interaction networks from the top left to the bottom right accordingly, and blank spaces represent zero values.

The transition matrix $N$, which describes transitions between layers, will take the form shown in Eq. 5.15. In this matrix, the layers are positioned in the same way as in the matrix $M$ (see the identical matrix borders).

The initial state vector $\mathbf{v}_0$ must be initialized and normalized. It will have two values since drug $l2$ is present in two layers: drug-similarity network and the bipartite layer. After the normalization process, the vector will be as shown in Eq. 5.16.

$$M_0 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccccccccc} t_1 & t_2 & t_3 & t_4 & t_1 & t_2 & t_3 & t_4 & l_1 & l_2 & l_3 & l_4 & l_5 & t_1 & t_2 & t_3 & t_4 & l_1 & l_2 & l_3 & l_4 & l_5 \end{array} \\ \left[\begin{array}{ccccccccccccccccccccc} 0 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & & \\ 1 & 0 & 1 & 1 & & & & & & & & & & & & & & & & & & \\ 1 & 1 & 0 & 1 & & & & & & & & & & & & & & & & & & \\ 1 & 1 & 1 & 0 & & & & & & & & & & & & & & & & & & \\ & & & & 0 & 1 & 1 & 1 & & & & & & & & & & & & & & \\ & & & & 1 & 0 & 1 & 1 & & & & & & & & & & & & & & \\ & & & & 1 & 1 & 0 & 1 & & & & & & & & & & & & & & \\ & & & & 1 & 1 & 1 & 0 & & & & & & & & & & & & & & \\ & & & & & & & & 0 & 1 & 1 & 1 & 1 & & & & & & & & & \\ & & & & & & & & 1 & 0 & 1 & 1 & 1 & & & & & & & & & \\ & & & & & & & & 1 & 1 & 0 & 1 & 1 & & & & & & & & & \\ & & & & & & & & 1 & 1 & 1 & 0 & 1 & & & & & & & & & \\ & & & & & & & & 1 & 1 & 1 & 1 & 0 & & & & & & & & & \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \tag{5.14}$$

$$N = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccccccccc} t_1 & t_2 & t_3 & t_4 & t_1 & t_2 & t_3 & t_4 & l_1 & l_2 & l_3 & l_4 & l_5 & t_1 & t_2 & t_3 & t_4 & l_1 & l_2 & l_3 & l_4 & l_5 \end{array} \\ \left[\begin{array}{ccccccccccccccccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & & & & & & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & & & & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & & & & & & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & & & & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & & & & & & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array} \tag{5.15}$$

$$\mathbf{v}_0{}^{norm} = \begin{array}{ccccccccccccccccccccc} t_1 & t_2 & t_3 & t_4 & t_1 & t_2 & t_3 & t_4 & l_1 & l_2 & l_3 & l_4 & l_5 & t_1 & t_2 & t_3 & t_4 & l_1 & l_2 & l_3 & l_4 & l_5 \end{array}^T \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \end{bmatrix} \tag{5.16}$$

As already mentioned, the product of $M$ and $N$ can be precomputed to simplify the following computations. After the normalization, the resulting transition matrix will be as in Eq. 5.17.

$$(M_0 \cdot N)^{norm} =$$

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $l_1$ | $l_2$ | $l_3$ | $l_4$ | $l_5$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $l_1$ | $l_2$ | $l_3$ | $l_4$ | $l_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $t_1$ | 0 | 1/9 | 1/9 | 1/6 | 0 | 1/9 | 1/9 | 1/6 | | | | | | 0 | 1/9 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_2$ | 1/6 | 0 | 1/9 | 1/6 | 1/6 | 0 | 1/9 | 1/6 | | | | | | 0 | 0 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | 1/6 | 1/9 | 0 | 1/6 | 1/6 | 1/9 | 0 | 1/6 | | | | | | 0 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_4$ | 1/6 | 1/9 | 1/9 | 0 | 1/6 | 1/9 | 1/9 | 0 | | | | | | 0 | 1/9 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_1$ | 0 | 1/9 | 1/9 | 1/6 | 0 | 1/9 | 1/9 | 1/6 | | | | | | 0 | 1/9 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_2$ | 1/6 | 0 | 1/9 | 1/6 | 1/6 | 0 | 1/9 | 1/6 | | | | | | 0 | 0 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | 1/6 | 1/9 | 0 | 1/6 | 1/6 | 1/9 | 0 | 1/6 | | | | | | 0 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_4$ | 1/6 | 1/9 | 1/9 | 0 | 1/6 | 1/9 | 1/9 | 0 | | | | | | 0 | 1/9 | 1/9 | 0 | 0 | 0 | 0 | 0 | 0 |
| $l_1$ | | | | | | | | | 0 | 1/8 | 1/4 | 1/4 | 1/8 | 0 | 0 | 0 | 0 | 0 | 1/8 | 0 | 0 | 1/8 |
| $l_2$ | | | | | | | | | 1/4 | 0 | 1/4 | 1/4 | 1/8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/8 |
| $l_3$ | | | | | | | | | 1/4 | 1/8 | 0 | 1/4 | 1/8 | 0 | 0 | 0 | 0 | 0 | 1/8 | 0 | 0 | 1/8 |
| $l_4$ | | | | | | | | | 1/4 | 1/8 | 1/4 | 0 | 1/8 | 0 | 0 | 0 | 0 | 0 | 1/8 | 0 | 0 | 1/8 |
| $l_5$ | | | | | | | | | 1/4 | 1/8 | 1/4 | 1/4 | 0 | 0 | 0 | 0 | 0 | 0 | 1/8 | 0 | 0 | 0 |
| $t_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/4 | 0 | 0 | 0 |
| $t_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/4 | 0 | 0 | 1/2 | 0 | 0 | 0 | 0 | 0 | 1/4 | 0 | 0 | 1/2 |
| $t_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $l_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $l_2$ | 0 | 1/3 | 1/6 | 0 | 0 | 1/3 | 1/6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/3 | 1/6 | 0 | 0 | 0 | 0 | 0 | 0 |
| $l_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $l_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $l_5$ | 0 | 0 | 1/6 | 0 | 0 | 0 | 1/6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/6 | 0 | 0 | 0 | 0 | 0 |

(5.17)

Using the precomputed matrix, the first iteration will get a result as shown in Eq. 5.18.

$$\mathbf{v}_1 = (M_0 \cdot N)^{norm} \cdot \mathbf{v}_0 =$$

| $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $l_1$ | $l_2$ | $l_3$ | $l_4$ | $l_5$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $l_1$ | $l_2$ | $l_3$ | $l_4$ | $l_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/8 | 0 | 1/8 | 1/8 | 1/8 | 0 | 1/4 | 1/4 | 0 | 0 | 0 | 0 | 0 | 0 |

$^T$

(5.18)

The iteration process lasts until the difference between the current state vector and one computed on the previous step is below the selected threshold. In our example, the last iteration will be on step 16, when the sum of absolute difference between values of $\mathbf{v}_{16}$ and $\mathbf{v}_{15}$ becomes less than $10^{-3}$. After the merging step the resulting vector $\mathbf{v}_{16}$ will be the one in Eq. 5.19.

According to the resulting vector, the random walker started from $l2$ will end its walk in $t1$, $t2$, $t3$ or $t4$ with probabilities 0.1174, 0.1569, 0.1956 and 0.1174 respectively. We ignore $t2$ and $t3$, since these vertices are already linked to $l2$. Once we do that, we will have $t1$ and $t4$ left with equal probabilities 0.1174. This means that for $l2$ links with both of these vertices are possible with equal probability. The final result is positive: the interaction between $l2$ and $t4$ is possible, because its probability is higher than given threshold of 0.1.

$$\mathbf{v}_{16} = (M_0 \cdot N)^{norm} \cdot \mathbf{v}_{15} = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \end{matrix} \begin{bmatrix} 0.0587 \\ 0.0608 \\ 0.0565 \\ 0.0587 \\ 0.0587 \\ 0.0608 \\ 0.0565 \\ 0.0587 \\ 0.0591 \\ 0.0562 \\ 0.0591 \\ 0.0591 \\ 0.0620 \\ 0 \\ 0.0353 \\ 0.0826 \\ 0 \\ 0 \\ 0.0848 \\ 0 \\ 0 \\ 0.0326 \end{bmatrix} \Longrightarrow \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \end{matrix} \begin{bmatrix} 0.1174 \\ 0.1569 \\ 0.1956 \\ 0.1174 \\ 0.059 \\ 0.141 \\ 0.059 \\ 0.059 \\ 0.0946 \end{bmatrix} \tag{5.19}$$

To conclude, this example demonstrates that NEWERMINE is able to solve the link prediction task on multi-layer graph with more than 3 layers, the main limitation of the state of the art approaches (Section 5.1.1).

## 5.3 Conclusion

We have developed an approach, NEWERMINE, for exploiting an arbitrary number of layers combined into a multi-layer network, proposing general matrix formulations to form intra- and inter-network transitions.

As EXPLICIT RANDOM WALKER from Chapter 4, which was designed to solve the same task, NEWERMINE has possibility of result interpretability. The prediction can be backtracked in the fashion of a classical walker, and feedback can be generated based on the visited nodes and their connections. However, comparing to EXPLICIT RANDOM WALKER, NEWERMINE is deterministic, i.e. its results are not changing over each run. In addition, the shown example and the results of NEWERMINE evaluation presented further in Chapter 6 demonstrate efficiency of the latter in comparison to classical approaches based on random walk. Finally, the result provided by NEWERMINE is already normalized, and due to the nature of the matrix computations there is no need to repeat the walk multiple times, both of which make NEWERMINE more efficient.

# 6. Experimental evaluation of random walk methods

*In this chapter, I present and discuss the experimental results of the* NEWERMINE *approach developed in Section 5.2 to solve the link prediction problem defined in Chapter 1. I start with the description of the evaluation framework. I then present the results with different settings and compare them with the ones of the* EXPLICIT RANDOM WALKER *defined in Chapter 4 and used as a baseline. Next, present details and discuss results of the sparsification experiment, which is performed to demonstrate that adding layers to the artificially sparsified bipartite layer should help to improve performance of our method. Finally, I perform validation of link predictions founded on external knowledge using the framework described in Chapter 3. I then finish the chapter with the conclusion.*

## 6.1 Experimental settings

In this section, I describe the evaluation framework. This includes the description of the evaluation protocol, the quality measures used, and followed by implementation details.

### 6.1.1 Evaluation Protocol

We perform evaluation of NEWERMINE, our approach developed in Section 5.2, on the IUPHAR data set described in Section 2.1.1. We also use EXPLICIT RANDOM WALKER, our approach defined in Section 4.2.1, to make a comparison with NEWERMINE. Because of its ease of implementation and high computational cost we use EXPLICIT RANDOM WALKER as a baseline.

To perform evaluation of our methods, *leave-one-out* cross-validation is used, which is defined as follows. For each of the 12456 edges in the bipartite layer of IUPHAR, we remove the edge from the network, set the drug as the starting vertex, and compute the path strengths for all possible drug-target paths for the given drug. Then we check whether the removed edge is present in the top remaining paths w.r.t. their strengths by using Precision at 20 (defined in [97]) and ignoring drug-target edges contained in the training data. If this is the case for an interacting edge, it is considered a *true positive*, otherwise a *false negative*. For negative examples (see Section 2.1.1 for more details), the relationship is inverse: *true negative* if removed edge is predicted correctly, and *false positive* otherwise.

Regarding the baseline, due to the randomness, a single run of EXPLICIT RANDOM WALKER may not be representative. We therefore infer path strengths by repeating the walk $k$ times. Any time the walker reaches *any* target vertex, we store the drug-target path with its current strength.

At the end of the $k$ walks, we compute the maximum of the strength values for all targets, which we then use to rank the paths.

Since NEWERMINE is deterministic, it is not necessary to repeat the iterations several times. In addition, there is no need to collect intermediate data, since we have strengths for the paths to all vertices in the network at the end of a single run.

### 6.1.2 Quality Measures

To evaluate our methods, we use several common performance measures: accuracy, precision, recall and AUC (Section 1.1.3). In addition to this, we report weighted versions of accuracy, precision, and recall which give more accurate assessments for unbalanced data sets as in our case.

We define weighted quality measures as follows. Due to the fact that the number of negative examples are smaller than that of the positives in our data (Section 2.1.1), we assign a classification cost of 1 to positive examples and cost *neg_cost* to negative ones, derived as:

$$neg\_cost = \frac{|D|}{2 \times |N|}, \tag{6.1}$$

where $|D|$ is the total number of examples, and $|N|$ is the number of negative examples. We then perform evaluation based on the costs defined: FN and TN receive score *neg_cost* for every negative example w.r.t. its real class, while FP and TP receives score 1 for positives.

### 6.1.3 Implementation

I implemented both EXPLICIT RANDOM WALKER and NEWERMINE in Python. I used the following libraries: networkx to model the multi-layer network, NumPy to perform all matrix computations and sklearn for cost-based evaluation.

## 6.2 Experimental results

In this section, I present the results using different combinations of three-layer networks and the full graph with six layers for NEWERMINE. I then compare the results on the latter setting with EXPLICIT RANDOM WALKER, the approach from Chapter 4 used as a baseline in this experiment.

### 6.2.1 Using three-layer graphs

We first use NEWERMINE on a number of multi-layer graphs aggregated from three networks each: one drug-drug network, one target-target network, and the drug-target network. This is the setting used in the state of the art discussed in Section 5.1.1.

For the experiments, we defined 6 possible combinations of three-layer networks with the bipartite layer of IUPHAR using the networks described in Section 2.1.1:

1. *DB* + *BG*: DrugBank + BioGrid + IUPHAR,
2. *DB* + *TS (ss)*: DrugBank + Target similarity (substrings) + IUPHAR,
3. *DB* + *TS (mot)*: DrugBank + Target similarity (motifs) + IUPHAR,
4. *DS* + *BG*: Drug similarity + BioGrid + IUPHAR,
5. *DS* + *TS (ss)*: Drug similarity + Target similarity (substrings) + IUPHAR,
6. *DS* + *TS (mot)*: Drug similarity + Target similarity (motifs) + IUPHAR.

The basic properties of the combinations compared to the full graph with six layers are presented in Table 6.1. In addition to basic indicators such as numbers of vertices and edges we report sparsity (Def. 1.1.13 in Chapter 1) and the number of connected components (Def. 1.1.22 in Chapter 1).

The first three combinations of networks are very sparse and highly disconnected (the first three rows in Table 6.1). Drug similarity networks added to the bipartite layer improve this situation (rows from four to six in Table 6.1). When *all* networks are combined with the bipartite layer,

Table 6.1: Basic properties of different combinations of networks

| Combination | Drugs | Targets | $\|V\|$ | $\|E\|$ | Sparsity | CC |
|---|---|---|---|---|---|---|
| *DB + BG* | 7025 | 2307 | 9332 | 143922 | 0.003 | 87 |
| *DB + TS (ss)* | 7025 | 2101 | 9126 | 1786917 | 0.042 | 103 |
| *DB + TS (mot)* | 7025 | 2101 | 9126 | 1786917 | 0.042 | 103 |
| *DS + BG* | 8056 | 2307 | 10363 | 23280724 | 0.434 | 21 |
| *DS + TS (ss)* | 8056 | 2101 | 10157 | 24923719 | 0.4832 | 22 |
| *DS + TS (mot)* | 8056 | 2101 | 10157 | 24923719 | 0.4832 | 22 |
| All six layers | 8137 | 2502 | 10639 | 26706838 | 0.4719 | 1 |

Table 6.2: Evaluation results of NEWERMINE with different combinations of three-layer graph

| Combination | Performance | | | | | | | Steps** |
|---|---|---|---|---|---|---|---|---|
| | Acc | Acc* | AUC | Prec | Prec* | Rec | Rec* | |
| *DB + BG* | 0.21 | 0.41 | 0.53 | 0.95 | 0.74 | 0.15 | 0.15 | 12 |
| *DB + TS (ss)* | 0.23 | 0.41 | 0.51 | 0.93 | 0.68 | 0.17 | 0.17 | 10 |
| *DB + TS (mot)* | 0.23 | 0.41 | 0.51 | 0.93 | 0.68 | 0.18 | 0.18 | 10 |
| *DS + BG* | 0.48 | 0.50 | 0.52 | 0.93 | 0.67 | 0.47 | 0.47 | 11 |
| *DS + TS (ss)* | 0.51 | 0.51 | 0.50 | 0.92 | 0.65 | 0.51 | 0.51 | 9 |
| *DS + TS (mot)* | 0.51 | 0.51 | 0.50 | 0.92 | 0.65 | 0.51 | 0.51 | 9 |

* weighted (cost-based)

** average (mean) to predict 1 link

it results in a single connected component (the bottom row of the table). The actual number of vertices in different networks depends on available IDs used for mapping and structural information used for computing similarities. In the case of the fully connected graph, the numbers of vertices and edges become higher than in any of the three-layer combination, which also results in a *lower* value of sparsity. In order to check if the actual sparsity of the fully connected network has been improved, we can recompute it for the last two combinations of three-layer setting, assuming that they have the same number of vertices as the full network with six layers (which is equal to adding disconnected vertices to the network, i.e. for which mapping cannot be found):

$$Sparsity(G_{(DS+TS)}) = \frac{2 \cdot 24923719}{10639 \cdot (10639-1)} = 0.4404.$$

The actual sparsity of the full network with six layers is still higher than this value, from which we can conclude that adding layers helped not only to connect the components, but also to improve the real sparsity of the network.

For the experiments, we use NEWERMINE with the following parameters: $\beta = 0.7$ recommended by the authors of NRWRH in [34] and $\eta = 0.2$ based on the recommended value in [34], taking into account that our parameter biases the walk in a reverse way, but at the same time that we do not employ the user specific parameter $\lambda$ responsible also for biasing the walk. This parameter setting is rather conservative, equivalent to relatively few steps before the walker restarts.

The results are presented in Table 6.2. The table shows that different three-layer graphs provide quite different results, but the main one is that drug similarity network computed using structural information and used instead of DrugBank significantly improves performance in terms of accuracy and recall (the last three rows compared to the first three rows in the table). This result correlates with the fact that the last three combinations are less sparse and better connected than others. In the next section, we check if the same holds for the fully connected graph with 6 layers.

### 6.2.2 Using the full graph with six layers

In this experiment, we evaluate NEWERMINE on the full graph with six layers. We use the same parameters as before: $\eta$=0.2, $\beta$=0.7. In addition, we test $\eta = 0$, which strongly biases the walk towards targets, and we also consider $\beta = 0.8$ for $\eta = 0.2$. The results are presented in Table 6.3.

Table 6.3: Evaluation results of NEWERMINE (NW) with different parameter settings and its comparison with EXPLICIT RANDOM WALKER (ERW) on the six-layer graph

| Approach | Performance | | | | | | | Steps** |
|---|---|---|---|---|---|---|---|---|
| | Acc | Acc* | AUC | Prec | Prec* | Rec | Rec* | |
| NW: $\eta = 0.2, \beta = 0.7$ | 0.47 | 0.54 | 0.57 | 0.95 | 0.73 | 0.45 | 0.45 | 9 |
| NW: $\eta = 0.2, \beta = 0.8$ | 0.45 | 0.53 | 0.57 | 0.95 | 0.73 | 0.43 | 0.43 | 10 |
| NW: $\eta = 0, \beta = 0.7$ | 0.48 | 0.54 | 0.58 | 0.95 | 0.74 | 0.47 | 0.47 | 8 |
| ERW: $\beta = 0.95$ | 0.35 | 0.49 | 0.56 | 0.95 | 0.74 | 0.31 | 0.31 | - |

* weighted (cost-based)

** average (mean) to predict 1 link

The results show that when more layers are used compared to the three-layer setting, recall drops (the last two combinations in Table 6.2 corresponding to best recall values vs. any of the parameter setting of NEWERMINE in Table 6.3). However, using the full six-layer network, weighted accuracy becomes higher (taking the lower proportion of negative examples into account), AUC score and precision do as well (any combination in Table 6.2 vs. results in Table 6.3).

Different parameter values do not have a large effect on the results but change the running time: increased $\beta$ also results in increased number of steps necessary for the iteration process to reach convergence, and decreased $\eta$ results in decreasing this number. The quality measure values are slightly better when using $\eta = 0$, however we will continue using value 0.2 in this work, since this parameter helps the walker to make his moves in the situation when the network is not fully connected. Biasing the walk towards targets helps to avoid these situation (in case of $\eta = 0$ the walker might not arrive in some parts if they are disconnected, which will produce zero ranking for vertices in such components).

We also run experiments with EXPLICIT RANDOM WALKER with the following parameters selected arbitrarily: $k = 1000$[1], $max\_steps = 100$, $\beta = 0.95$. Note, $\beta$ used in EXPLICIT RANDOM WALKER has the same meaning as $\beta$ in NEWERMINE, but due to the algorithmic difference, it is implemented differently, which also results in the selection of different value of $\beta$. The results are presented in Table 6.3 (the bottom row).

NEWERMINE with any of the tested parameters clearly outperforms EXPLICIT RANDOM WALKER in terms of both weighted and unweighted accuracy, weighted and unweighted recall, and AUC. Moreover, EXPLICIT RANDOM WALKER is a much more expensive algorithm to run in terms of time. In the given setting, it requires 1000 times to repeat the walk[1], while NEWERMINE comes to convergence in 10 steps in average, which is roughly 100 times faster than EXPLICIT RANDOM WALKER.

To conclude, NEWERMINE provides better results than the relatively simple baseline approach. In addition, its result becomes better when the number of networks is higher and the data is not sparse and fully connected. In the next section, we check the last point one more time on the artificially sparsified network.

[1] Repeating the walk fewer number of times, e.g. $k = 10$ or 100, might not be enough for the walker to find paths between all querying drug-target pairs in the given network.

## 6.3 Sparsification experiment

In this section, I present results of the sparsification experiment performed on the IUPHAR data set. This includes the experiment description and is followed by the results and a discussion.

### 6.3.1 Idea of the experiment

By doing this experiment, we would like to demonstrate that adding networks to the bipartite layer actually helps to improve performance of our method. For achieving that, we artificially sparsify the IUPHAR network step by step by removing equivalent portions of edges from the bipartite layer. We then run experiments on two settings: the bipartite layer of IUPHAR only and the full network with 6 layers. According to our hypothesis, adding layers should help to make the network more dense and less disconnected, and that should result in a better performance.

In total, we perform 10 experiments with different proportions of links present in the bipartite layer of IUPHAR: from 10% to 100% with a step size of 10 percentage points. We run each experiment twice: for the IUPHAR only setting and the fully connected graph with 6 layers. To be able to better capture the difference between different proportions of links, we removed existing negative links from the bipartite layer and assume all non-existing links in that layer to be negative. In this new configuration, the number of positive and negative examples became highly unbalanced and we had to change the evaluation framework to one using curves, the ROC and PR-curves in particular, with evaluation by AUC and AUPR as quality measures. It also resulted in changing the evaluation protocol to the one using *5x5 stratified cross-validation*, defined as follows. We randomly split non-existing (negative) links into 5 folds, and perform the same for existing (positive) links. Then we set each fold with negative links and each with positive ones to be *test set* while the combination of the rest 8 folds to be *learning set*. The prediction is performed on learning set and evaluation on test set. The process is repeated 5 times, and the results are averaged among all runs. To make predictions, we run NEWERMINE with the recommended parameters: $\eta = 0.2$ and $\beta = 0.7$ (Section 6.2.2).

The basic properties of networks for both settings and all subsets of links are presented in Table 6.4. We use all vertices for learning and test, 10639 to be more precise (not shown in the table), and thus this value remains the same for each combination of each network setting and percentage of links. It is also important to point out that there is no setting with one single connected component even when 100% of links are used. This be explained by the fact that removing negative links from the bipartite layer made some parts of the network unreachable, which in turn resulted in a relatively high number of disconnected components.

Table 6.4: Basic network properties of the IUPHAR only network and the full graph with 6 layers and different % of links in the bipartite layer

| % of links | IUPHAR network | | | 6 networks graph | | |
|---|---|---|---|---|---|---|
| | $\|E\|$ | Sparsity | $CC$ | $\|E\|$ | Sparsity | $CC$ |
| 10 | 3216 | 0.00006 | 7772 | 26697598 | 0.47178 | 816 |
| 20 | 4135 | 0.00007 | 7129 | 26698517 | 0.47180 | 700 |
| 30 | 5054 | 0.00009 | 6535 | 26699436 | 0.47181 | 598 |
| 40 | 5973 | 0.00011 | 5999 | 26700355 | 0.47183 | 497 |
| 50 | 6892 | 0.00012 | 5498 | 26701274 | 0.47185 | 404 |
| 60 | 7811 | 0.00014 | 5039 | 26702193 | 0.47186 | 320 |
| 70 | 8730 | 0.00015 | 4596 | 26703112 | 0.47188 | 242 |
| 80 | 9649 | 0.00017 | 4187 | 26704031 | 0.47190 | 169 |
| 90 | 10568 | 0.00019 | 3800 | 26704950 | 0.47191 | 99 |
| 100 | 11487 | 0.00020 | 3436 | 26705869 | 0.47193 | 34 |

### 6.3.2 Results and discussion

The results are presented in Fig. 6.1 and 6.2 for AUC and AUPR respectively.

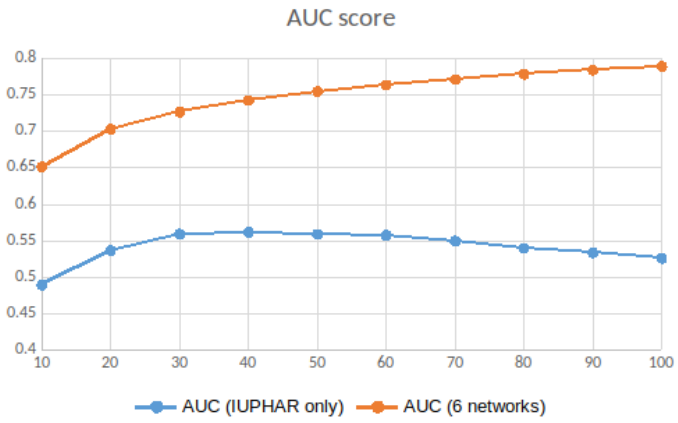


Figure 6.1: Results of the experiment on sparsification of IUPHAR (AUC score)

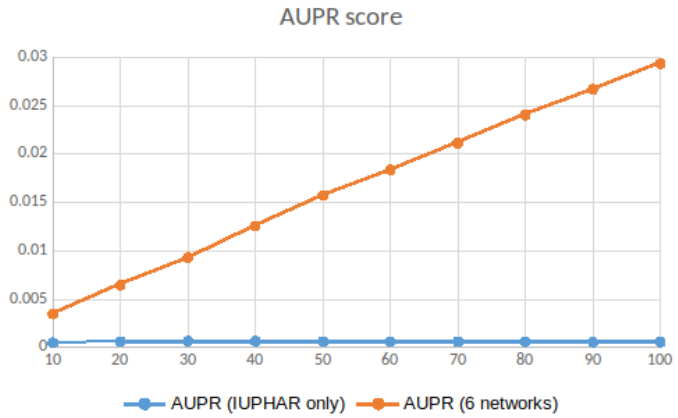


Figure 6.2: Results of the experiment on sparsification of IUPHAR (AUPR score)

As was expected, additional layers in the 6 layer setting helped to improve AUC and AUPR for all the proportions of links compared to the bipartite only setting. Surprisingly, the AUC for the IUPHAR only network drops when more than 40% of the links are used, until at 100% it is roughly equal to the one for 20%. (Fig. 6.1). On the other hand, the maximum AUC of around 0.79 that was reached with 100% of links is quite high compared to the results from Section 6.2.2, where the maximum AUC was not exceeding 0.58. This difference can be explained by two factors: different evaluation protocols which were used in these experiments and also different configurations of the data. The principle difference of the first is that $5\times5$ cross-validation with validation by curves used in the sparsification experiment learns the threshold to determine class labels of test examples while the older setting, leave-one-out cross-validation in combination with Precision at 20, uses a fixed threshold of top 20 predicted interactions to decide on class labels of assessing examples, which is a more selective measure. Concerning the configuration of the data, in the sparsification experiment we did not consider negative examples. However, according to our previous results shown in Table 6.3, NEWERMINE is better performing in predicting existing positive links than existing negative ones, since its precision was reported as high (0.95), but the overall accuracy was not that high. This fact also positively impacted on the performance results in the sparsification experiment.

## 6.4 Validation of link prediction with external knowledge

In this section, I present results of evaluation of NEWERMINE using the UMLS as an external resource, the setting described in Chapter 3.

### 6.4.1 Experimental setup

In this experiment, we evaluate the quality of predictions made by NEWERMINE. Predictions are performed on the full 6 layer graph of IUPHAR, and the quality of predictions is assessed by using the validation framework presented in Chapter 3. We use the full data i.e. without splitting into train and test sets, and predict edges that are not present in IUPHAR *at all*. Predicted interactions are then evaluated on the UMLS.

As before, we use NEWERMINE with the parameter setting selected in Section 6.2.2 ($\eta = 0.2$, $\beta = 0.7$) to run the algorithm and Precision at 20 to predict new interactions. Running that for each drug in the set results in a constant number of predictions: $8137 \times 20 = 162740$, where 8137 is the number of drugs in IUPHAR (Section 2.1.1).

We use the *2018AB* release[2] of the UMLS, which contains 14,438,243 concept names related to distinct 3,822,832 concepts, presented by 155 sources in 25 different languages. As explained in Chapter 3, the UMLS is not complete and we proposed a solution for that. As explained in that chapter, to get more realistic results, we estimate an upper bound of maximum response. For that, we use all drugs and targets from IUPHAR, even those for which we do not have available properties (and thus not used in the main experiments). To specify our solution regarding the current experiment, using all drugs and targets gives us $8978 \times 2910$ combinations of drugs and targets respectively minus 12456 existing interactions in the denominator of maximum response equation (Eq. 3.14).

For evaluating the results, we apply our framework described in Section 3.2.1, which is tested with the four different similarity measures defined in Section 3.1.3. In this framework, we search for occurrences of predicted results in the UMLS, use predefined confidence score and threshold, and, finally, give a feedback. As quality measures, precision and normalized precision defined in Eq. 3.12, 3.13 are reported.

### 6.4.2 Experimental results

The Non-zero score has the highest precision, validating the majority of predictions (almost 69%), while the Semantic vector similarity based score does lowest giving positive feedback only to a tiny subset (less then 0.01%) of predictions (first column in Table 6.5). By definition, the Non-zero score provides with the maximum number of verified links, and the Semantic vector score is the most selective measure. The result of the former is too optimistic, because the structure of the hierarchy is not taken into account by the measure, while the latter captures the maximum information about the hierarchy of the concepts associated with the predicted interaction pair. Path and Lin similarity based scores, in their turn, are intermediately positioned in the amount of information they are able to capture from the hierarchy and the results they demonstrate (first column of Table 6.5).

In addition to non-normalized values, Table 6.5 reports normalized version of precision given by the UMLS as defined in Section 3.2.2. To compute that, we first determine maximum response given by the UMLS defined in Eq. 3.14 with relation to each similarity measure from Section 3.1.3.

Maximum response w.r.t. different measures is shown in Table 6.6. Trend in maximum response values remains the same as for values of non-normalized precision for same measures. The reason for that is the same as before: the Non-zero score is the most inclusive, while the Semantic vector score is the most exclusive in this comparison.

[2]Released on 25/09/2018

Table 6.5: Percentage of verified predictions made by NEWERMINE using the UMLS with different similarity measures

| Measure | *Prec* % | $Prec^{norm}$ % |
|---|---|---|
| Non-zero score | 68.88 | 88.11 |
| Path similarity | 19.07 | 81.07 |
| Lin similarity | 0.99 | 85.31 |
| Semantic vector | 0.01 | 10.48 |
| median | | 83.19 |

Table 6.6: Maximum response of IUPHAR on the UMLS using different similarity measures

| Measure | Max response % |
|---|---|
| Non-zero score | 78.17 |
| Path similarity | 23.52 |
| Lin similarity | 1.16 |
| Semantic vector | 0.07 |

The normalization process balances the differences between different measures by adjusting 3 out of 4 to approximately same range of values (second column in Table 6.5). Their median shown in the bottom row of Table 6.5 supports this observation. The resulting value of normalized precision of about 83% demonstrates good result. According to this estimation, more than 80% of the interactions predicted by NEWERMINE can be verified using the UMLS as an external resource, taking into account limitations of the UMLS. This result with the vast majority of verified interactions led us to a conclusion that it is rather possible to use NEWERMINE to make predictions on a new data.

## 6.5 Conclusion

We have demonstrated experimentally on real and artificially sparsified data that our approach developed in the previous chapter works, and that combining different networks actually improves vertex reachability and therefore interaction prediction. The comparison with relatively simple baseline showed better performance of our approach in terms of both prediction accuracy and computational cost.

In addition, we performed an extensive evaluation of our method on the external knowledge showing its effectiveness on predicting new unseen interactions. The results we received using cross-validation on labeled data are comparable to ones using our framework based on external resource evaluating new unlabeled data, with majority of verified interactions in both cases. Thereby, we can conclude that optimal parameter settings can be transferred to a new unseen data.

# 7. Analysis of connectivity characteristics

*The results of the previous chapter showed that link prediction in networks works better when those networks are connected and not sparse. The purpose of this chapter is to try to find experimentally if common connectivity characteristics can be used to determine the quality of networks before the link prediction process starts. I start this chapter with providing a more detailed motivation. I then present experimental settings we use to perform experiments. This is followed by the experimental evaluation, which includes the description of connectivity characteristics used, and, finally, I finish with the conclusion. The experimental results of this work are the main contribution of this chapter, also published at GEM'19.*

## 7.1 Motivation

The motivation for this work comes from the previous chapter where we predicted the likelihood that a drug and a target interact, based on the information of their other interactions, and the interactions of other drugs and targets. The network built of known interactions between drugs and targets is often sparse and not fully connected, limiting how well additional interactions can be predicted. To improve this, some methods in the literature propose adding additional information to original data, e.g. about entities' similarities or interactions of entities of the same type. The typical configuration in the literature consists of a three-layer network: the interaction layer, as well as one each connecting drugs to drugs and targets to targets, respectively (Section 1.2.1). To make this configuration, some choices are to be made about which similarity measure to use or which additional networks to use or discard.

Our hypothesis in Chapter 6 was a simple one: instead of making these choices, we would use all available networks and let the prediction algorithm handle this situation. The random walk algorithm we proposed, NEWERMINE, did exactly that on a large and sparse interaction network, IUPHAR: as we showed experimentally, running NEWERMINE on a six-layer network from IUPHAR and the five additional networks which are available gave better results to using *any* three-layer network we could construct. We associated this result with the fact that only the six-layer network formed a fully connected component while having the same sparsity as some of the better-performing three-layer networks.

Further results, however, questioned this assumption. Performing similar experiments on benchmark data sets introduced in Section 2.1.2, we found out that using only the three layers selected by the authors of the sets, NEWERMINE systematically performed better than on six-layer

or eight-layer networks.

There are at least two possible explanations for this. According to the first, some of the networks do not provide reliable information, or the second, once the network has a certain structure that is sufficient for the success of the random walk, adding additional layers does not help anymore. To explore this problem in more detail, we present a systematic study of how combining different networks into multi-layer graphs affects connectivity characteristics of the resulting network, and whether we can use those characteristics to explain the quality of the resulting link predictions.

## 7.2 Experimental settings

### 7.2.1 The data

As mentioned in the introduction, we perform our experiments on benchmark data sets described in Section 2.1.2: Enzime, GPCR, IC, NR and Kinase. The original representation of these data consists of three networks: a drug similarity network obtained by the use of the SimComp score that we will refer to as DS(sc), a target similarity network obtained by the use of the Smith-Waterman score (TS(sw)), and a drug-target interaction network (DT) constructed from the KEGG BRITE [87], BRENDA [145], SuperTarget [70], and DrugBank [163] databases. To build eight-layer networks[1] for our experiments, we have augmented this representation by constructing additional networks using the same principle as for the IUPHAR network described in Section 2.1.1:

- a drug-drug interaction network based on DrugBank (DB),
- a target-target interaction network based on BioGrid [150] (BG),
- a drug similarity network based on similarities calculated using the Tanimoto coefficient on binary vectors constructed from the presence/absence of frequent subgraphs (DS(sg)), and
- two target similarity networks calculated using the Tanimoto coefficient on feature vectors constructed from the presence/absence of *frequent substrings* (TS(ss)) and *Prosite motifs* (TS(mot)).

Drugs were mapped between networks by numerical identifiers provided by KEGG. Targets were mapped by KEGG identifiers as well as by Human Entrez Gene attribute. Additional similarity networks were constructed using the same approach as in Section 2.1.1, but instead of SMILES codes, a graph representation of drug molecules was used directly, which was parsed from KEGG[2] (for Enzyme, GPCR, IC and NR) and NCBI databases[3] (for Kinase), and the amino-acid sequences for targets were collected by also parsing KEGG (for Enzyme, GPCR, IC and NR) and NCBI (for Kinase) databases.

As in case of IUPHAR, not all drugs and targets contained in benchmark sets are available in DB and BG. In addition, not all proteins and drugs are annotated with molecular information so that not all entities will be connected in the similarity networks.

### 7.2.2 Experimental protocol

As a prediction method we use NEWERMINE, the random-walk based method we have proposed in Section 5.2, with recommended parameter settings from Section 6.2 ($\eta = 0.2$, $\beta = 0.7$).

Unlike the evaluation framework used in Section 6.2, we performed a *5x5-fold cross-validation*, with each fold containing 20% of all drug-target interactions, used as test set for link prediction once, while the method is run on the other 80%. The process is repeated 5 times, the results are averaged among all runs. We switched to a 5x5-fold cross-validation to be able to compare the results with state-of-the-art methods.

[1] the maximum number of layers that could be achieved combining all available sources of information

[2] https://www.genome.jp/kegg/

[3] https://pubchem.ncbi.nlm.nih.gov

To assess the quality of prediction we report both area under the ROC curve (AUC) and area under the precision-recall curve (AUPR) introduced in Section 1.1.3.

## 7.3 Experimental evaluation

As we wrote above, our previous results in Chapter 6 indicated that using all available networks leads to better performance than using the common three network setting. In this section, we first explore whether the same behavior holds for the benchmark data sets. We then explore how many and which combinations of networks allow for best performance before discussing the connectivity characteristics of those best-performing combinations. Finally, we discuss the connectivity characteristics and evaluate how they transfer to the IUPHAR data.

### 7.3.1 Link prediction in benchmark data sets

Table 7.1 reports results of NEWERMINE on the benchmark data sets for using both the three layers proposed in the literature, and the eight layers that result from augmenting those networks with the additional information we described above in Section 7.2.1.

As we can see from table 7.1, using all layers clearly underperforms compared to using only the original three, the common configuration in the state of the art. This result is surprising, specially taking into account that having more data in a learning or mining setting is usually better. Trying to find an answer to this issue we can formulate two hypotheses providing possible explanations:

1. **Certain networks are less informative than others.**
   DrugBank information, for instance, might be less helpful since interactions are based on published results in the literature, whereas similarity edges are more informative: the molecular information about a compound exists and has been translated into a similarity value in a transparent manner. But not all similarities are equal: the SimComp score takes chemical aspects into account that the use of frequent subgraphs might miss.
2. **There are diminishing returns to adding additional layers.**
   To arrive at a network that can be exploited for link predictions by a random walk process (such as NEWERMINE), it is necessary to create as few connected components as possible, and a network that is as dense as possible. But once the entire network consists of a single connected component, and paths between any two vertices are short, it might be impossible to improve the state of the network.

Table 7.1: Link prediction results of NEWERMINE on the benchmark data using three-layer and eight-layer networks

| Data set | Performace | | | |
|---|---|---|---|---|
| | 3 layer network | | 8 layer network | |
| | AUC | AUPR | AUC | AUPR |
| Enzyme | 0.84 | 0.15 | 0.66 | 0.15 |
| GPCR | 0.81 | 0.22 | 0.67 | 0.11 |
| IC | 0.76 | 0.26 | 0.61 | 0.09 |
| NR | 0.64 | 0.26 | 0.49 | 0.15 |
| Kinase | 0.61 | 0.13 | 0.55 | 0.08 |

One of these properties (or a combination of both) can have a negative impact on a random walk process. Similar to how adding noisy attributes or ones that are strongly correlated with attributes that are already present can degrade the performance of a predictor for vector data, adding non-reliable or redundant edges can destabilize link prediction. In the next sections, we will evaluate both of our hypotheses.

### 7.3.2 Connectivity characteristics and redundancy

We start with exploring the second hypothesis from Section 7.3.1. Table 7.2 illustrates the main characteristics of the different data sets and we can see that IUPHAR differs from the other sets: significantly less dense, with a much higher number of distinct connected components. Notably, the Kinase network is also different from other benchmark sets since it is the only one that is already connected (the number of connected components is equal to 1).

Table 7.2: Basic properties of benchmark and IUPHAR data sets

| Data set | Drugs | Targets | $\|V\|$ | $\|E\|$ | Sparsity | CC |
|---|---|---|---|---|---|---|
| Enzyme | 445 | 664 | 1109 | 2926 | 0.0048 | 44 |
| GPCR | 223 | 95 | 318 | 635 | 0.0126 | 19 |
| IC | 210 | 204 | 414 | 1476 | 0.0173 | 3 |
| NR | 54 | 26 | 80 | 90 | 0.0285 | 10 |
| Kinase | 65* | 373* | 438* | 1527 | 0.0160 | 1 |
| IUPHAR | 6580* | 1454* | 8034* | 12456 | 0.0004 | 443 |

* ignoring isolated nodes, a property of our implementation

And as Table 7.3 shows, adding two additional networks boosts the density of the benchmark data sets and turns them into a single connected component, whereas even the best 3-network combination for IUPHAR shown in the table still results in a sparse, fragmented network. The reason for this is that even in the case of the rich similarity networks there are missing connections, the fact we mentioned before. The reader will also notice that some network configurations indicate fewer vertices than in the original data sets. This happened because isolated vertices, i.e. those not connected to any other vertex in any layer, were not taken into consideration due to our implementation.

Table 7.3: Basic properties of three-layer networks of benchmark and IUPHAR

| Data set | Drugs | Targets | Layers | $\|V\|$ | $\|E\|$ | Sparsity | CC |
|---|---|---|---|---|---|---|---|
| Enzyme | 445 | 664 | 3 | 1109 | 321832 | 0.5238 | 1 |
| GPCR | 223 | 95 | 3 | 318 | 29853 | 0.5923 | 1 |
| IC | 210 | 204 | 3 | 414 | 44127 | 0.5162 | 1 |
| NR | 54 | 26 | 3 | 80 | 1846 | 0.5842 | 1 |
| Kinase | 68 | 442 | 3 | 510 | 101266 | 0.7802 | 1 |
| IUPHAR | 7025* | 2010* | 3 | 9126* | 1786917 | 0.0215 | 103 |

* ignoring isolated nodes, a property of our implementation

This is probably a first indication for why using three layers for the IUPHAR network is not enough – it does not explain, however, why adding additional layers decreases the quality of link prediction on the benchmark data sets.

To gain more insight into this question, we present a number of connectivity characteristics of derived multi-layer networks in Tables 7.4 – 7.8. For better readability, we do not show the complete list of network combinations but only representative results. The first column lists which additional networks have been added to the DT network (which is always present and therefore omitted). The tables then list, in order,

- the number of layers that the network has ($|L|$),
- its sparsity, i.e. the number of existing edges divided by the maximum possible number of edges (see Def. 1.1.13 in Chapter 1 for more details),

Table 7.4: Connectivity characteristics and performance of different network combinations (each including DT layer) for the Enzyme data set

| Network combination | Connectivity characteristics | | | | | | | | | Performance | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|L\|$ | Sparsity | $CC$ | Giant component | | | Flattened graph | | | AUC | AUPR |
| | | | | $r$ | $d$ | Avg dist | $\|E\|$ | $C_{clust}$ | $C_{trans}$ | | |
| BG | 2 | 0.0068 | 12 | 6 | 10 | 4.3650 | 4153 | 0.1599 | 0.0562 | 0.24 | 0.01 |
| DB | 2 | 0.0177 | 14 | 5 | 10 | 3.5028 | 10856 | 0.2127 | 0.4350 | 0.66 | 0.10 |
| DS(sg) | 2 | 0.1656 | 1 | 2 | 3 | 2.1675 | 101716 | 0.7298 | 0.9710 | 0.82 | 0.14 |
| DS(sc) | 2 | 0.1656 | 1 | 2 | 3 | 2.1675 | 101716 | 0.7298 | 0.9710 | 0.82 | 0.14 |
| **DS(sg),TS(sw)** | **3** | **0.5238** | **1** | **2** | **2** | **1.4762** | **321832** | **0.9825** | **0.9838** | **0.84** | **0.15** |
| ***DS(sc),TS(sw)*** | ***3*** | ***0.5238*** | ***1*** | ***2*** | ***2*** | ***1.4762*** | ***321832*** | ***0.9825*** | ***0.9838*** | ***0.84*** | ***0.15*** |
| DS(sg),DB,TS(sw),TS(ss) | 5 | 0.8950 | 1 | 2 | 2 | 1.4762 | 321832 | 0.9825 | 0.9838 | 0.74 | 0.09 |
| DS(sg),DB,TS(sw),TS(mot) | 5 | 0.8950 | 1 | 2 | 2 | 1.4762 | 321832 | 0.9825 | 0.9838 | 0.74 | 0.09 |
| DS(sc),DB,TS(sw),TS(ss) | 5 | 0.8950 | 1 | 2 | 2 | 1.4762 | 321832 | 0.9825 | 0.9838 | 0.74 | 0.10 |
| DS(sc),DB,TS(sw),TS(mot) | 5 | 0.8950 | 1 | 2 | 2 | 1.4762 | 321832 | 0.9825 | 0.9838 | 0.74 | 0.09 |

Table 7.5: Connectivity characteristics and performance of different network combinations (each including DT layer) for the GPCR data set

| Network combination | Connectivity characteristics | | | | | | | | | Performance | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|L\|$ | Sparsity | $CC$ | Giant component | | | Flattened graph | | | AUC | AUPR |
| | | | | $r$ | $d$ | Avg dist | $\|E\|$ | $C_{clust}$ | $C_{trans}$ | | |
| BG | 2 | 0.0129 | 14 | 6 | 12 | 4.4993 | 650 | 0.0251 | 0.0078 | 0.28 | 0.02 |
| DB | 2 | 0.0868 | 7 | 5 | 9 | 2.8299 | 4369 | 0.3706 | 0.5779 | 0.65 | 0.19 |
| DS(sg) | 2 | 0.5037 | 1 | 2 | 3 | 1.5716 | 25388 | 0.8771 | 0.9764 | 0.77 | 0.22 |
| DS(sc) | 2 | 0.5037 | 1 | 2 | 3 | 1.5716 | 25388 | 0.8771 | 0.9764 | 0.77 | 0.22 |
| **DS(sg),TS(sw)** | **3** | **0.5923** | **1** | **2** | **2** | **1.4077** | **29853** | **0.9546** | **0.9694** | **0.80** | **0.21** |
| ***DS(sc),TS(sw)*** | ***3*** | ***0.5923*** | ***1*** | ***2*** | ***2*** | ***1.4077*** | ***29853*** | ***0.9546*** | ***0.9694*** | ***0.80*** | ***0.22*** |
| DS(sg),DB,TS(sw),TS(ss) | 5 | 0.7550 | 1 | 2 | 2 | 1.4077 | 29853 | 0.9546 | 0.9694 | 0.73 | 0.17 |
| DS(sg),DB,TS(sw),TS(mot) | 5 | 0.7550 | 1 | 2 | 2 | 1.4077 | 29853 | 0.9546 | 0.9694 | 0.74 | 0.17 |
| DS(sc),DB,TS(sw),TS(ss) | 5 | 0.7550 | 1 | 2 | 2 | 1.4077 | 29853 | 0.9546 | 0.9694 | 0.74 | 0.18 |
| DS(sc),DB,TS(sw),TS(mot) | 5 | 0.7550 | 1 | 2 | 2 | 1.4077 | 29853 | 0.9546 | 0.9694 | 0.74 | 0.18 |

Table 7.6: Connectivity characteristics and performance of different network combinations (each including DT layer) for the IC data set

| Network combination | Connectivity characteristics | | | | | | | | | Performance | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|L\|$ | Sparsity | $CC$ | Giant component | | | Flattened graph | | | AUC | AUPR |
| | | | | $r$ | $d$ | Avg dist | $\|E\|$ | $C_{clust}$ | $C_{trans}$ | | |
| BG | 2 | 0.0185 | 3 | 5 | 9 | 3.7715 | 1583 | 0.0958 | 0.0340 | 0.32 | 0.02 |
| DB | 2 | 0.0780 | 3 | 4 | 7 | 2.5661 | 6666 | 0.3639 | 0.6033 | 0.63 | 0.14 |
| DS(sg) | 2 | 0.2740 | 1 | 2 | 3 | 1.8648 | 23421 | 0.9169 | 0.9367 | 0.71 | 0.16 |
| DS(sc) | 2 | 0.2740 | 1 | 2 | 3 | 1.8648 | 23421 | 0.9169 | 0.9367 | 0.72 | 0.17 |
| **DS(sg),TS(sw)** | **3** | **0.5162** | **1** | **2** | **2** | **1.4838** | **44127** | **0.9469** | **0.9414** | **0.76** | **0.26** |
| ***DS(sc),TS(sw)*** | ***3*** | ***0.5162*** | ***1*** | ***2*** | ***2*** | ***1.4838*** | ***44127*** | ***0.9469*** | ***0.9414*** | ***0.76*** | ***0.27*** |
| DS(sg),DB,TS(sw),TS(ss) | 5 | 0.8191 | 1 | 2 | 2 | 1.4838 | 44127 | 0.9469 | 0.9414 | 0.67 | 0.15 |
| DS(sg),DB,TS(sw),TS(mot) | 5 | 0.8191 | 1 | 2 | 2 | 1.4838 | 44127 | 0.9469 | 0.9414 | 0.67 | 0.15 |
| DS(sc),DB,TS(sw),TS(ss) | 5 | 0.8191 | 1 | 2 | 2 | 1.4838 | 44127 | 0.9469 | 0.9414 | 0.68 | 0.15 |
| DS(sc),DB,TS(sw),TS(mot) | 5 | 0.8191 | 1 | 2 | 2 | 1.4838 | 44127 | 0.9469 | 0.9414 | 0.67 | 0.16 |

Table 7.7: Connectivity characteristics and performance of different network combinations (each including DT layer) for the NR data set

| Network combination | Connectivity characteristics | | | | | | | | | Performance | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|L\|$ | Sparsity | $CC$ | Giant component | | | Flattened graph | | | AUC | AUPR |
| | | | | $r$ | $d$ | Avg dist | $\|E\|$ | $C_{clust}$ | $C_{trans}$ | | |
| BG | 2 | 0.0434 | 2 | 4 | 6 | 3.4679 | 137 | 0.2241 | 0.2318 | 0.26 | 0.04 |
| DB | 2 | 0.1073 | 3 | 3 | 5 | 2.4042 | 339 | 0.2717 | 0.3026 | 0.50 | 0.10 |
| DS(sg) | 2 | 0.4813 | 1 | 2 | 3 | 1.6044 | 1521 | 0.8648 | 0.9452 | 0.61 | 0.22 |
| DS(sc) | 2 | 0.4813 | 1 | 2 | 3 | 1.6044 | 1521 | 0.8648 | 0.9452 | 0.63 | 0.28 |
| DS(sg),TS(sw) | 3 | 0.5842 | 1 | 2 | 2 | 1.4158 | 1846 | 0.9096 | 0.9295 | 0.60 | 0.20 |
| *DS(sc),TS(sw)* | *3* | *0.5842* | *1* | *2* | *2* | *1.4158* | *1846* | *0.9096* | *0.9295* | *0.63* | *0.26* |
| DS(sg),BG | 3 | 0.4962 | 1 | 2 | 3 | 1.5535 | 1568 | 0.7892 | 0.9413 | 0.66 | 0.19 |
| **DS(sc),BG** | **3** | **0.4962** | **1** | **2** | **3** | **1.5535** | **1568** | **0.7892** | **0.9413** | **0.69** | **0.22** |
| DS(sg),DB,TS(sw),TS(ss) | 5 | 0.7658 | 1 | 2 | 2 | 1.4158 | 1846 | 0.9096 | 0.9295 | 0.51 | 0.13 |
| DS(sg),DB,TS(sw),TS(mot) | 5 | 0.7658 | 1 | 2 | 2 | 1.4158 | 1846 | 0.9096 | 0.9295 | 0.48 | 0.12 |
| DS(sc),DB,TS(sw),TS(ss) | 5 | 0.7658 | 1 | 2 | 2 | 1.4158 | 1846 | 0.9096 | 0.9295 | 0.53 | 0.15 |
| DS(sc),DB,TS(sw),TS(mot) | 5 | 0.7658 | 1 | 2 | 2 | 1.4158 | 1846 | 0.9096 | 0.9295 | 0.52 | 0.16 |

- the number of connected components (*CC*), where by *connected component* we assume a graph any two vertices of which are connected to each other by a path, and which is connected to no additional vertices in the supergraph (see Def. 1.1.22 in Chapter 1 for more precise definition),
- the radius of the network, i.e. the *minimum* shortest path between any two different vertices in the network (*r*),
- its diameter, the *maximum* shortest path between any two different vertices (*d*),
- average distance between vertices, computed as mean of shortest paths between all not equal vertices of the network (Avg dist),
- the number of edges of the flattened graph ($|E|$),
- the clustering coefficient ($C_{clust}$), the number of existing closed triangles divided by the maximum possible number of open and close triangles, and
- the transitivity ($C_{trans}$), the number of existing closed triangles divided by the number of connected triples.

Finally, the right-most column lists the AUC and AUPR scores for link prediction via NEWERMINE. It should be noted that connectivity characteristics such as clustering coefficient and transitivity are typically not defined for multi-layer networks – we therefore report these measures for a flattened network: when there is an edge available between any two nodes, they are treated as connected regardless the network the edge belongs to. It should also be mentioned that in networks having more than one connected component, we compute and report radius, diameter and average distance values for the largest such component, called *giant*, in terms of the number of vertices. For a network with the number of $CC = 1$, its giant component is equal to the network itself.

It can be seen from Tables 7.4 – 7.7 that reducing the number of connected components improves link prediction, especially once only a single connected component remains. For all of these benchmark data sets, the latter is achieved at the latest once the network contains three layers.

A secondary effect can be noticed once the diameter of the network is decreasing to the value of 2: for Enzyme, GPCR, and IC, the final improvement in AUC happens once the network moves from a diameter of 3 to one of 2. This decrease in the diameter appears at the same time with an increase in the value of the clustering coefficient. Notably, however, better results on NR are achieved for a diameter of 3 than one of 2.

Once those two requirements – the number of connected components and the diameter – have been satisfied, there is no further improvement from adding additional networks. Moreover, making the multi-layer network significantly denser *does not* result in additional improvements but instead reduces quality.

### 7.3.3 Informativeness of networks

We suppose that this regression in link prediction quality, when the optimal point is passed, is related to the first hypothesis that we formulated above in Section 7.3.1 – certain networks simply provide better information.

Tables 7.4 – 7.8 also show that not all multi-layer networks that lead to similar or even the same connectivity characteristics allow for the same performance of NEWERMINE. In each table, we have indicated the best-performing network combination (or combinations) in **bold** and the original network combination – using SimComp for drug similarity and Smith-Waterman for target similarity – in *italics*.

As we can see, for Enzyme, GPCR, and IC, the original combination actually gives best results. However, while Smith-Waterman seems to encode vital information about target similarity, the SimComp score can be replaced by a very simple similarity measure, the Tanimoto coefficient computed on the vectors of frequent subgraphs. In other words, subgraphs mined in an unsupervised manner can provide information that is not worse compared to a similarity measure based on more

Table 7.8: Connectivity characteristics and performance of different network combinations (each including DT layer) for the Kinase data set

| Network combination | Connectivity characteristics | | | | | | | | | Performance | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|L\|$ | Sparsity | $CC$ | Giant component | | | Flattened graph | | | AUC | AUPR |
| | | | | $r$ | $d$ | Avg dist | $\|E\|$ | $C_{clust}$ | $C_{trans}$ | | |
| BG | 2 | 0.0209 | 1 | 4 | 6 | 2.9244 | 2433 | 0.1447 | 0.0459 | 0.36 | 0.04 |
| DB | 2 | 0.0169 | 1 | 5 | 10 | 3.2539 | 1615 | 0.0363 | 0.0285 | 0.56 | 0.13 |
| **DS(sg)** | **2** | **0.0381** | **1** | **2** | **4** | **2.3750** | **3700** | **0.7136** | **0.5108** | **0.66** | **0.17** |
| DS(sc) | 2 | 0.0392 | 1 | 2 | 3 | 2.3718 | 3805 | 0.7319 | 0.5451 | 0.65 | 0.16 |
| DS(sg),DB | 3 | 0.0390 | 1 | 2 | 4 | 2.3750 | 3701 | 0.7137 | 0.5111 | 0.65 | 0.16 |
| DS(sc),DB | 3 | 0.0401 | 1 | 2 | 3 | 2.3718 | 3805 | 0.7319 | 0.5451 | 0.64 | 0.14 |
| DS(sg),BG | 3 | 0.0391 | 1 | 3 | 6 | 2.4816 | 4606 | 0.5234 | 0.4938 | 0.65 | 0.12 |
| DS(sc),BG | 3 | 0.0400 | 1 | 3 | 6 | 2.4791 | 4711 | 0.5320 | 0.5272 | 0.64 | 0.11 |
| DS(sg),TS(sw) | 3 | 0.7794 | 1 | 2 | 3 | 1.2222 | 101161 | 0.9540 | 0.9847 | 0.61 | 0.14 |
| *DS(sc),TS(sw)* | *3* | *0.7802* | *1* | *2* | *3* | *1.2214* | *101266* | *0.9578* | *0.9848* | *0.61* | *0.13* |
| DS(sg),DB,TS(sw),TS(ss) | 5 | 1.5159 | 1 | 2 | 3 | 1.2222 | 101162 | 0.9541 | 0.9847 | 0.58 | 0.12 |
| DS(sg),DB,TS(sw),TS(mot) | 5 | 1.5159 | 1 | 2 | 3 | 1.2222 | 101162 | 0.9541 | 0.9847 | 0.58 | 0.12 |
| DS(sc),DB,TS(sw),TS(ss) | 5 | 1.5167 | 1 | 2 | 3 | 1.2214 | 101266 | 0.9578 | 0.9848 | 0.58 | 0.11 |
| DS(sc),DB,TS(sw),TS(mot) | 5 | 1.5167 | 1 | 2 | 3 | 1.2214 | 101266 | 0.9578 | 0.9848 | 0.58 | 0.11 |

Table 7.9: Connectivity characteristics and performance of different network combinations (each including DT layer) for the IUPHAR data set

| Network combination | Connectivity characteristics | | | | | | | | | Performance |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\|L\|$ | Sparsity | $CC$ | Giant component | | | Flattened graph | | | AUC |
| | | | | $r$ | $d$ | Avg dist | $\|E\|$ | $C_{clust}$ | $C_{trans}$ | |
| DS(sg) | 2 | 0.5147 | 69 | 5 | 9 | 1.7761 | 23272066 | 0.7992 | 0.9996 | 0.5781 |
| DB, BG | 3 | 0.0033 | 87 | 7 | 14 | 4.8069 | 143922 | 0.1285 | 0.5118 | 0.5255 |
| DS(sg), DB, TS(ss) | 4 | 0.4780 | 14 | 5 | 9 | 1.7714 | 24929849 | 0.9003 | 0.9992 | 0.5805 |
| DS(sg), DB, TS(mot) | 4 | 0.4780 | 14 | 5 | 9 | 1.7714 | 24929849 | 0.9003 | 0.9992 | 0.5800 |
| DS(sg), DB, TS(ss), TS(mot) | 5 | 0.5095 | 14 | 5 | 9 | 1.7714 | 24929849 | 0.9003 | 0.9992 | 0.5812 |
| all | 6 | 0.4719 | 1 | 5 | 9 | 1.8587 | 24932609 | 0.8802 | 0.9992 | 0.5735 |

detailed properties of its entities.

For the other two data sets, NR and Kinase, the results are even more interesting: using the original layer combination for the first does *not* give best results – instead combining BioGrid information about target-target interactions with SimComp drug similarity results in the highest AUC. The reason for that could be that NR is too small in terms of the number of vertices and the number of edges for the evaluation framework we use (when we remove 20% of edges the network might become too disrupted resulting in such unpredictable behavior). And there is an explanation for Kinase also. As we already mentioned above, Kinase is different from other benchmark sets, particularly because it already consists of a single connected component. Adding the subgraph-based drug similarity layer DS(sg) to the interaction network is enough to achieve the highest AUC, even higher than one provided by the original combination with DS(sc), TS(sw) networks and a number of others that do not use the TS(sw) network at all.

### 7.3.4 Comparison with IUPHAR

After having explored how connectivity characteristics on benchmark data sets correlate with the quality of link prediction, we would like to check if similar behavior can be observed in our initial problem setting introduced in Chapter 6 – link prediction on the IUPHAR data set.

In Table 7.9, we list the connectivity characteristics of the *best-performing* network combinations for IUPHAR, one for each number of layers from two to six. There is an exception for four layers where the AUC was virtually indistinguishable. We have limited the maximum number of layers to six since there is no DS(sc) nor TS(sw) network available for the IUPHAR set.

There are a number of interesting observations in our results.

1. The complete set of experiments conducted in this chapter shows that the best layer combination is five (while experiments in Section 6.2 could suggest the best layer combination contain all six layers), the BioGrid layer seems to reduce the quality of the information in the multi-layer network.
2. The second is that the importance of reducing radius and diameter hold – 5 and 9 are the two lowest values we can achieve on IUPHAR for radius and diameter respectively, and combinations corresponding to them are better performing compared to the 3-layer setting with higher radius and diameter values.
3. The third observation is that the exception to this rule is the three-layer setting – this setting, typical in the literature, seems to be the single worst setting for IUPHAR and the improvement on it is therefore easiest.
4. Fourth, once radius and diameter have been minimized, the number of connected components seems to become less important, a phenomenon we could not observe on the smaller benchmark data sets.
5. Finally, there were several other combinations that achieved a radius equal to 5 and a diameter to 9, without leading to best AUC. The difference is that better performing combinations have lower average distance between vertices.

## 7.4 Conclusion

We did not test all layer combinations for all networks for running time reasons, and did not find any strong correlations between connectivity characteristics and the quality measures. However, we have many experimental results to draw some conclusions based on them. First of all, when augmenting networks for link prediction, it is important to add data that reduce the radius, diameter and average distance of the final network as much as possible.

Another conclusion we can draw is that the five benchmark data sets that are being widely used in the state of the art on drug-target activity prediction might not be the best for the given

problem setting. They are relatively small, can easily be augmented to form only a single connected component and achieve minimal diameter. For the Kinase data set, it is in particular true since it has a single connected component from a single drug-target interaction layer.

Similarly to how data sets in other fields have become outdated and are rarely used nowadays – e.g. the Iris data set for machine learning and clustering, or the Mutagenicity data set for inductive logic programming and substructure mining – it might be time to find new benchmarks and replace these networks with them. As such, the IUPHAR network proposed in this thesis can be one of these new benchmarks.

# III

# Predicting links with communities


**8 Link prediction via community detection . . . . . 83**
8.1 Related work
8.2 The LPBYCD approach
8.3 Conclusion

**9 Experimental evaluation of predicting links with communities . 101**
9.1 Experimental setup
9.2 Experimental results
9.3 Validation of link prediction with external knowledge
9.4 Conclusion

# 8. Link prediction via community detection

*In this chapter, I address the link prediction problem defined in Chapter 1 focusing on methods exploiting community information. I first present related work which includes the state of the art on link prediction using communities, link prediction measures used in the state of the art and community detection methods themselves. I then present the* LPBYCD *approach, an adaptation of existing techniques to our problem setting defined in Chapter 1. This includes the adaptation of different types of communities to the bipartite setting, the adaptation of existing link prediction measures to the bipartite and/or multi-layer setting and a systematic technique for automatic selection of optimal community detection parameters. I then finish the chapter with the conclusion. The* LPBYCD *approach is the main contribution in this chapter, also published at SAC'20.*

## 8.1 Related work

In this section, I present the state of the art about community detection in general and describe how communities can be used for link prediction. I start with a literature overview on how community information can be exploited to perform link prediction in general, and follow that by a summary on community detection in multi-layer graphs, the setting defined in Section 1.2.2. I then present link prediction measures and explain how they can be used for link prediction. Next, I present two common community detection methods, both of which are used further in Chapter 9 to test our approach developed in Section 8.2.

### 8.1.1 State of the art

The idea to use community information to predict links in graphs is not novel. There are several notable works exist on this topic.

#### Link prediction with communities

Clauset *et al.* in [40] proposed to exploit a learned hierarchical generative community model to estimate the probabilities of missing links in partially known networks. The authors of [69, 159, 172] combined community detection with existing edge prediction methods to improve prediction accuracy. These methods are based on the hypothesis that vertices in the same community have similar properties, and missing edges are more likely to be found within communities than elsewhere.

At the same time, the authors of [108] demonstrated that density of links between two communities can be exploited by a naïve Bayes model.

In [172] missing edges are predicted by node similarity using nearest-neighbor measures. In that method, proximity measures based on vertex neighborhoods are used for calculating the probabilities for pairs of nodes belonging to the same community. The authors of [69] use Stochastic Block Models to identify both missing and false edges. In a Stochastic Block Model, vertices are partitioned into groups and the probability that two nodes are connected depends only on the groups to which they belong. These groups of nodes can be understood as communities. While many approaches consider that vertices belong to just one community, in [159] missing edges are predicted by in-group/out-group neighbor similarity measures based on participation in multiple communities. Several new measures are proposed for that based on common neighbors within and outside of a common community, as well as with total and partial overlapping of communities. Edges can be predicted for vertices belonging to the same community even if there is no path between them within the community [82]. The authors of the latter consider meta-path-based features and examine the influence of the path length on the link prediction problem by employing a specific type of meta-path, originated from and ending at communities.

Communities can also be used the other way around. The authors of [1] reimagine communities as groups of edges rather than vertices. In that work, they solve the problem community detection approaches based on hierarchical clustering have in common: these approaches cannot capture the relationships between overlapping communities. According to the authors, link communities naturally incorporate these overlaps while disclosing hierarchical organization. At the same time, the authors of [148] use community detection to modify similarity measures. They show experimentally that community membership information can provide valuable information for link prediction.

Finally, a number of approaches are based on the assumption that two vertices are more likely to form a link if their sets of neighbors are overlapping in so called shared neighborhoods. There is a set of methods which extend the concept of shared neighborhoods [101] to community neighborhoods [30, 50, 168]. The latter three propose their own measures exploiting that assumption with relation to communities. In addition, in [79] neighborhood measures have been extended to multiple layers. The authors of that work redefine the notion of neighborhood to multi-layer neighborhood and propose new measures using an updated notion.

#### Community detection in multi-layer graphs

Community detection in multi-graphs can also be performed in different ways: directly, by ensemble-based methods, or by graph flattening.

The direct methods perform discovery of communities on the multi-layer network directly, e.g. by adapting objective functions for community detection to the multi-layer setting [47, 93, 154]. In [93] a random walk on the network is implemented. Based on this random walk, a node dissimilarity measure is derived and nodes are clustered in a hierarchical fashion. The proposed measure is used as a distance function to perform the clustering. The authors of [154] propose an extension of the modularity measure used to quantify the quality of communities to multi-layer networks. The updated modularity takes into account the layer-specific contributions of edges in the internal and external connectivity of the communities. In another work [47], the authors define interdependence between layers by measuring how much information about one layer helps to predict links in another one. This allows them to bundle layers together and identify small groups of layers which are able to accurately predict the remaining layers. They implement the method of maximum likelihood optimization to identify those groups of layers by optimizing the expected number of layer edges in each layer. The main drawback of the direct methods is implementation complexity, making them impossible to use "out of the box".

The second group of methods, ensemble-based, perform community detection on each layer separately, and aggregate discovered communities afterwards [154]. Their main uncertainty is

the aggregation mechanism, which requires additional optimization. For instance in [154], the aggregation of layer-specific community structures is performed as in clustering ensemble methods. The resulting clustering is computed by optimizing special modularity function, which is defined over information available from the different clusters in the ensemble.

Finally, flattening approaches summarize multiple edges into single ones and use the resulting single-layer network to discover communities by using one of the common community detection approaches such as spectral partitioning [98] or the Louvain algorithm [20]. In this work, we use the last type of approach, because it is able to provide the desired result, it is the easiest to use and its computational complexity is potentially the lowest.

### 8.1.2 Link prediction measures

We divide existing link prediction measures into two broad categories: neighborhood measures and others, which we refer to as community-based. We start by formally defining the measures using the graph notations introduced in Section 1.1.2.

#### Neighborhood measures

Neighborhood measures are based on the notion of neighborhood, i.e. the set of vertices directly connected to the examined vertices. Many of these measures exploit the neighborhoods of vertices in the form of *common neighbors*, the *Jaccard coefficient*, *preferential attachment*, or *SimRank* [101]:

1. The Common neighbors of two vertices $x$ and $y$ are defined as the intersection between their neighbors and counts how many neighbor vertices they have in common:

$$CN(x,y) = |\Gamma(x) \cap \Gamma(y)|, \tag{8.1}$$

   with $\Gamma(x)$ denoting the set of neighbors of $x$ formally defined by:

$$\Gamma(x) = \{v \mid (x,v) \in E\}. \tag{8.2}$$

2. The Jaccard coefficient extends common neighbors by normalizing its number by the total number of neighbors two vertices have:

$$JC(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}. \tag{8.3}$$

3. Preferential attachment is equal to the probability that $x$ and $y$ are neighbors, which correlates with the product of the number of neighbors both vertices have:

$$PA(x,y) = |\Gamma(x)| \cdot |\Gamma(y)|, \tag{8.4}$$

   with $|\Gamma(x)|$ denoting the number of neighbors of $x$.
4. SimRank, finally, measures the similarity of two vertices. According to this measure, two nodes are considered to be similar if they are referenced by similar vertices. Using the definition from [101], we can define a simplified form of SimRank as the number of common neighbors two vertices share normalized by the probability that they are neighbors:

$$SR(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| \cdot |\Gamma(y)|}. \tag{8.5}$$

The semantic similarity between neighborhood and community, i.e. sets of vertices in both cases, allows us to use neighborhood measures in our setting.

### Community-based measures

Other measures proposed in the literature are not based on a notion of neighborhood, but on other metrics, and thus we group them into a separate category. The measures in this group exploit community information in some sense, which is why we call them community-based.

Due to the specificity of our setting, which is bipartite and multi-layer at the same time, we cannot use most of the measures proposed in the literature, and we thus report only measures which can be adapted. This includes *CAR-based* measures by Cannistraci *et al.* [30], Xie *et al.*'s measures, which we refer as *Neighboring community-based* [168] and *Community relevance* by Ding *et al.* [50]:

1. CAR-based measures are based on the assumption that two nodes are more likely to link together if their common neighbors are members of a strongly linked cohort. This cohort is called local community, and its links are called local community links (LCL). From all variations of CAR measures we use the *CAR-based common neighbors* and the *CAR-based Jaccard coefficient* defined as:

$$CCN(x,y) = CN(x,y) \cdot LCL(x,y), \tag{8.6}$$

$$CJC(x,y) = \frac{CCN(x,y)}{|\Gamma(x) \cup \Gamma(y)|} = JC(x,y) \cdot LCL(x,y), \tag{8.7}$$

   with LCL is defined as:

$$LCL(x,y) = \sum_{s \in \Gamma(x) \cap \Gamma(y)} \frac{|\gamma(s)|}{2},$$

   where $\gamma(s)$ is the *local community degree* of $s$ (the subset of neighbors of $s$ that are also common neighbors of $x$ and $y$).
2. The *Neighboring community-based* measure exploits connection of vertices to communities, summing over all communities. Formally it is defined as the probability that two nodes $x$ and $y$ are *adjoined*:

$$NCB(x,y) = \sum_{i=1}^{k} P(x \sim C_i) P(y \sim C_i) P(C_i), \tag{8.8}$$

   where $k$ is the number of discovered communities, and $P(x \sim C_i)$ denotes the probability that vertex $x$ adjoin to community $C_i$ defined in its turn as:

$$P(x \sim C_i) = \frac{|\{v \mid (x,v) \in E, v \in C_i\}|}{|\{v \mid v \in C_i\}|},$$

   and $P(C_i)$ is the probability that a randomly chosen vertex is in the community $C_i$:

$$P(C_i) = \frac{|\{v_j \mid v_j \in C_i\}|}{|V|}.$$

   Two nodes $x$ and $y$ are said to be *adjoined* if community $C$ exists to which they adjoin: $x \sim C$, $y \sim C$.
3. Community relevance measures, finally, extend neighborhood measures to neighborhoods of communities. We only use the *Community relevance Jaccard coefficient* which is defined as:

$$CRJC(x,y) = \frac{|CRJC(C(x)) \cap CRJC(C(y))|}{|CRJC(C(x)) \cup CRJC(C(y))|}, \tag{8.9}$$

with $CRJC(C(v))$ representing all vertices belonging to the community and all of their neighbors:

$$CRJC(C(v)) = \Gamma(C(v)) \cup \{v \mid v \in C(v)\}.$$

CRJC measures the probability that both communities $C(x)$ and $C(y)$ share common neighbors.

### 8.1.3 Community detection methods

In this work, we use two common community detection approaches: spectral partitioning and the Louvain algorithm.

#### Spectral partitioning

To gain an intuitive idea of communities and to understand how they can be detected using spectral methods, one needs to understand first the graph partitioning task.

**Graph partitioning** Graph partitioning is the task of assigning the vertices of a graph to different components approximately equal in size while minimizing the number of edges between components [98]. This can, for instance, be achieved by solving the minimum cut problem (Fig. 8.1).

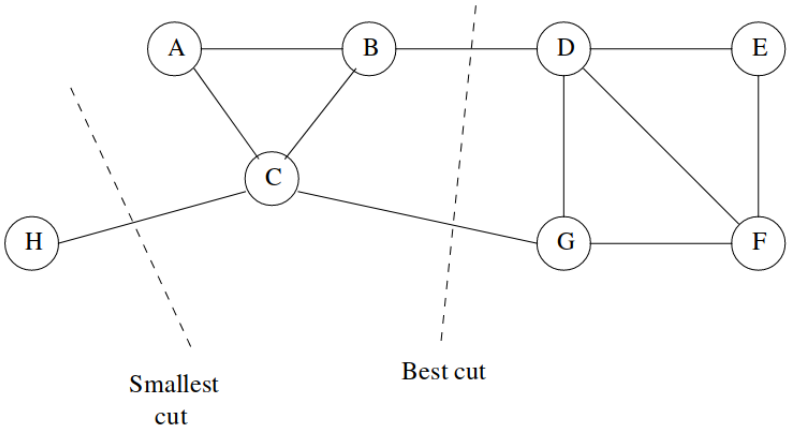


Figure 8.1: An example of a good partitioning of a graph: the smallest cut might not be the best [98]

The graph partitioning task can be solved using spectral methods, which we will summarize shortly. The spectral characteristics of algebraic matrices can be analyzed through their eigenvalues and eigenvectors, which can then be used to group different objects represented by the matrix. Those objects could be points in some metric space or the vertices of a graph as in our case. By projecting them into a space defined by the eigenvectors, one can use a properly defined splitting criterion to achieve partitions of desired sizes [56].

Spectral partitioning uses the Laplacian matrix (Def. 1.1.18), whose smallest eigenvalue is always 0, with all elements of the corresponding eigenvector 1 [98]. Moreover, as observed in [55], the eigenvector corresponding to the second smallest eigenvalue partitions the graph into two components via a minimum cut: vertices corresponding to positive elements of the eigenvector into one group, and those corresponding to negative ones into the other. Instead of this *sign cut* (which is equivalent to a threshold of 0), different thresholds can be used to perform the split. For instance, the *bisection cut* [67] uses the median value of entries in an eigenvector as a threshold, producing two components of approximately equal size. *Mean* can be used as an alternative for the bisection cut, which uses the average instead of median.

**Graph partitioning for community detection** Solving the graph partitioning task satisfies our definition of a community provided in Chapter 1. The resulting components produced as a solution to the minimum cut problem will have higher density of links inside components than between them that according to Def. 1.1.23 allows us to consider these components as communities.

Since for community detection we want more than two components, we have to use more than one eigenvector. Using the $m$ eigenvectors corresponding to the m smallest non-zero eigenvalues, a partition into at most $2^m$ groups can be found. Each eigenvector's values will be split according to the threshold, assigning a binary code (0 or 1) to each vertex. As a result, every vertex is labeled by a binary code in the range $[0, 2^m - 1]$, indicating its group membership. By definition, these groups do not overlap and can be considered as non-overlapping communities. Note that the maximum possible number of communities, $2^m$, is typically not achieved.

Using multiple eigenvectors means that thresholds can be derived in a number of ways. The same threshold can be applied to all eigenvectors, or each individual eigenvector can have its own threshold. Additionally, the threshold applied to all eigenvectors can be computed by applying the aggregating function (mean, median *etc.*) to *all* eigenvectors or only to the $m$ actually used.

### Louvain algorithm

The Louvain algorithm [20] belongs to the family of modularity-based community detection methods, and greedily optimizes modularity [56]. Its main advantage is the reduction of the $O(N^2)$ complexity of the greedy approach to linear complexity [12].

**Modularity** Modularity is a generic measure to determine the quality of each partition produced by a community detection approach [123]. The modularity of the partition is a scalar value between -1 and 1 that measures the density of links inside communities as compared to links between communities (Fig. 8.2).

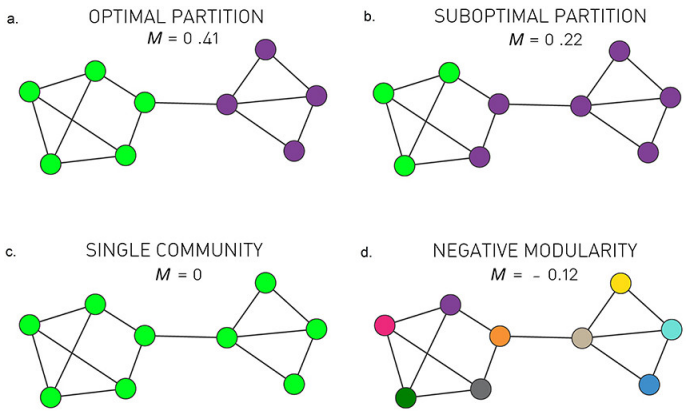


Figure 8.2: Modularity values and quality of discovered communities [12]

Modularity is defined as:

$$M_C = \frac{1}{2L} \sum_{v_i, v_j \in C} (A_{ij} - p_{ij}), \tag{8.10}$$

where $C$ is a community, $A_{ij}$ is the weight of the edge $(v_i, v_j)$, $p_{ij} = \frac{k_i k_j}{2L}$ its expected weight, $k_{(i)}$ the sum of the weights of $v_{(i)}$-adjacent edges, and $L = \frac{1}{2}\sum_{ij} A_{ij}$ [12, 20].

**Modularity optimization** The main process is similar to the one performed by agglomerative algorithms in hierarchical clustering, but using vertices instead of data points (Fig. 8.3).

The Louvain algorithm consists of two iteratively repeated phases or "passes". In the first phase, each vertex is considered a separate community. Next, the modularity gain of merging a

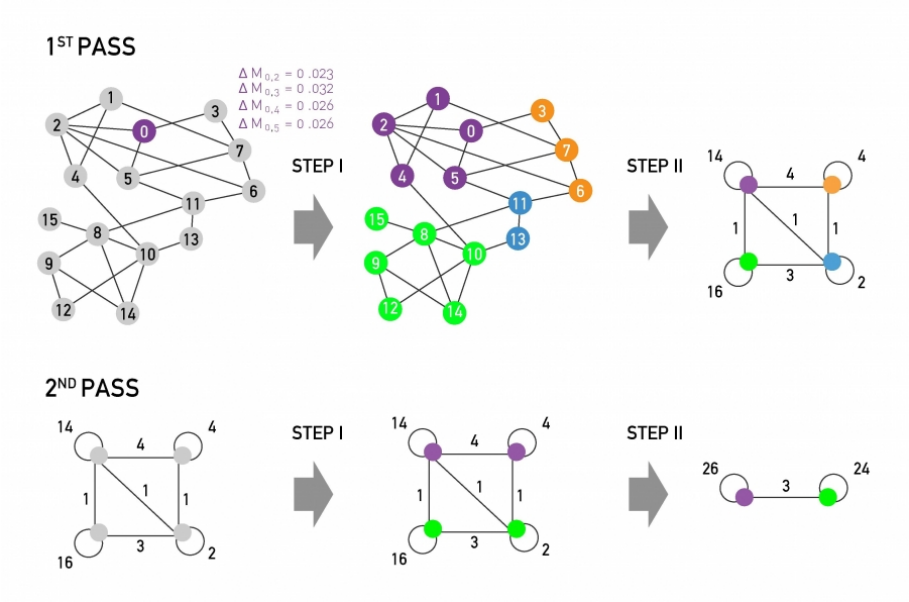


Figure 8.3: Steps and passes of the Louvain approach [12]

vertex with the community of one of its neighbors is evaluated, and the best merge is performed. In the second phase a new network is built whose vertices are the communities found during the first phase. Vertices are aggregated by summing up the weights of their edges. The process stops when maximum modularity has been reached. Modularity gain can be precomputed, making the algorithm rather efficient [12].

**Resolution parameter** The weak point of the Louvain algorithm is that it might fail when trying to identify communities smaller than a certain threshold. This threshold is called *resolution* and defines a modularity scale limit. Practically speaking, at different moments of time $t$ the difference between an optimal partitioning and the partitioning produced by Louvain is different and the resolution parameter represents this change in the time. For instance, at time $t = 0$ each of the nodes is in its own community, while at time $t \implies \infty$ the approach divides all the nodes into two communities [96]. The resolution limit parameter can be exploited to control the quality of results. It was noticed that with default value of $t = 1$ the algorithm produces decent partitions, however the quality of such partitioning can be improved by optimizing this parameter.

## 8.2 The LPBYCD approach

In our problem setting defined in Section 1.2.2, we want to predict links between two distinct types of nodes, e.g. drugs and targets. To achieve this, we perform community discovery using an existing community detection approach, then exploit discovered communities to solve the link prediction task. The resulting approach, that we call **L**ink-**P**rediction-**by**-**C**ommunity-**D**etection or LPBYCD, can be summarized as a 3 steps algorithm. On the first step of this algorithm, one needs to choose which community detection method to use, next to decide how the obtained communities will be exploited, finally to fix the link prediction measure (Fig. 8.4). In this section, I describe how communities are used with our approach (the second step of the algorithm) and explain how we adapt existing measures for link prediction to our setting (the third step). This is followed by the description of a systematic method for fixing the approach parameters, and we finish with an example.

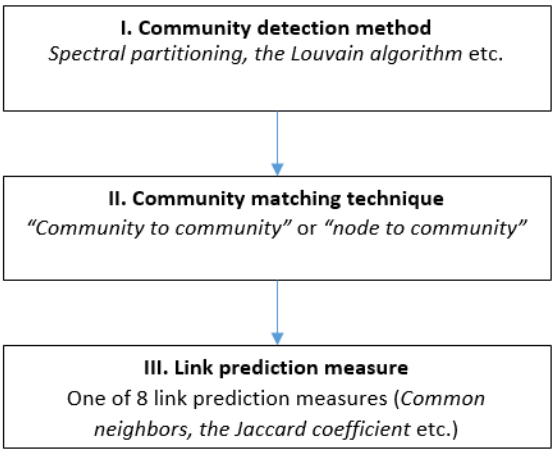


Figure 8.4: General schema of the LPBYCD approach

### 8.2.1 Link prediction in a bipartite setting

Due to the construction of the networks we use and the community detection methods we evaluate, resulting communities can be *mixed*, i.e. containing both types of nodes, drugs and targets, as well as *pure*, of either type, drugs or targets only (Fig. 8.5). Also, the community detection methods we use produce *non-overlapping* communities only, i.e. communities of which the nodes belong to one community at a time.

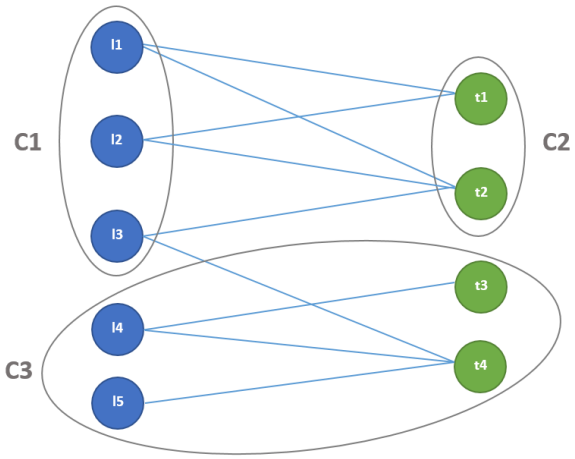


Figure 8.5: An example of *pure* and *mixed* communities: $C1$ and $C2$ are pure, and $C3$ is mixed

Mixed communities can be exploited directly with existing measures for link prediction via community detection (see Section 8.1.2), but pure communities cannot, ignoring a large number of drug-target pairs. To treat all communities in a consistent manner, we treat all of them as non-mixed and split mixed communities into pure ones (Fig. 8.6). Notably, this split does not have to be done explicitly, but a mixed community can be treated as two pure ones with links between them.

We exploit discovered communities in one of two proposed ways: by matching "community to community" or "node to community".

#### Community to community

In the first case, each drug community is paired with each target community, then an adapted measure is used to perform link prediction between paired communities (Fig. 8.7). Each non-

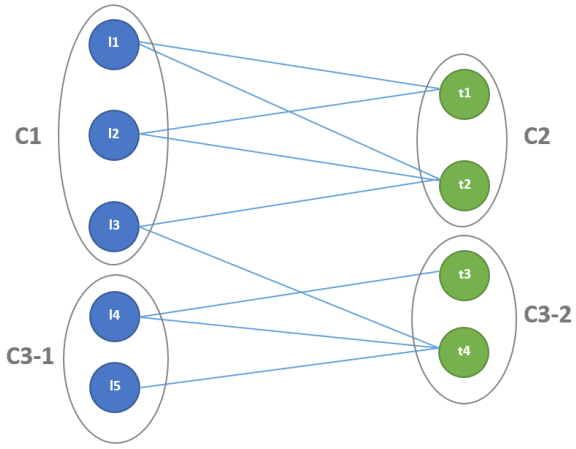


Figure 8.6: An example of splitting community $C3$ from Fig. 8.5 into 2 pure communities: $C3$-1 and $C3$-2

interacting drug-target pair between paired communities is assigned the same link probability score. At the end of the matching, each non-interacting drug-target pair from the network will have been assigned a single score, which can be used to rank predictions. We refer to this approach as *community to community* (or *CC*).

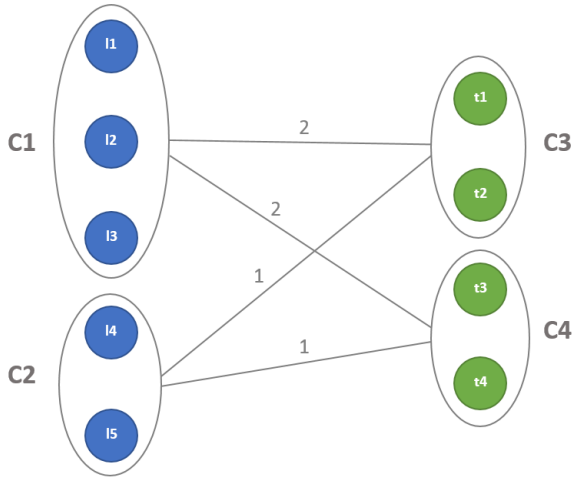


Figure 8.7: An example of *community to community* matching (label on edges represent number of existing edges between vertices of matched communities). Note that this example is independent from the network shown in Fig. 8.6

In the example shown in Fig. 8.7, community $C1$ is twice connected to community $C3$ and twice to community $C4$. $C2$, on the other hand, is connected once to $C3$ and once to $C4$. This matching therefore implicitly assumes that all ligands in $C1$ have a connection of strength two to all targets in $C3$ *etc.* The big advantage of this matching is that even vertices that have no bipartite connection at all can receive a positive score.

### Node to community

Another way of exploiting communities is to pair each node of one type with communities of the other type. The advantage of that method is that for a selected drug $d_i$ and target $t_j$ two predictions can be made: once analyzing connections of a drug with target communities and second

analyzing target connections with drug communities, providing a more reliable estimate (Fig. 8.8). This approach is also referred to as Bipartite Local Models [19].

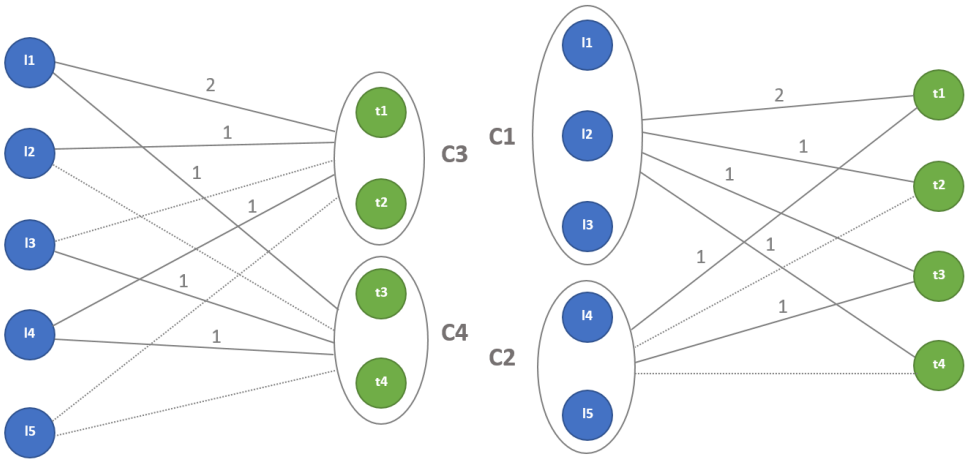


Figure 8.8: An example of *node to community* matching (labels on edges represent number of existing edges between vertices and matched communities, dotted edges represent non-interacting pairing)

Fig. 8.8 illustrates the difference of this method with the *CC* matching using the same communities and the same strengths of the connections as in Fig. 8.7. For example in Fig. 8.8, there is twice evidence for $l1$ and $t1$ to be connected with strength 2, twice evidence for $l2$ and $t2$ to be connected with strength 1, and mixed evidence for the connections between pairs $l1$ and $t2$, and $l2$ and $t1$ coming from different origin: $l1$ is strongly connected to the target community $C3$ with strength 2 and $t2$ is weakly connected to the drug community $C1$ with strength 1, and $l2$ is weakly connected to the target community $C3$, but $t1$ – strongly to the drug community $C1$. In the case of *CC* matching, only one evidence is provided with less detailed description: all combinations between $l1$, $t1$ and $t1$, $t2$ are strongly connected with strength 2 (Fig. 8.7).

Vertices belonging to the same community will therefore not necessarily receive the same score but a vertex such as $l5$ will be strongly punished because it is not connected with a bipartite edge. The link probability score between $d_i$ and $t_j$ is computed by aggregating the two results [28]. We report results using *mean* as an aggregation function. Our experiments showed that the difference between *max* and *mean* is negligible (see Table 1–3 in Appendix 1), and we use *mean* to get a more reliable result. We do not consider *min* as aggregator, because in the case of no evidence for existence of the link in one of the independent predictions the combined probability is also 0. We refer to this approach as *node to community* (or *NC*).

### 8.2.2 Adapting prediction measures

To be able to use the existing link-prediction-by-community-detection measures presented in Section 8.1.2, we have to adapt them to the bipartite setting, and to the multi-layer one where it is required. Note, that due to the characteristics of the data used in our experiments and for simplicity reasons we present our approach for the special case of a multi-layer graph with the number of bipartite layers $l = 1$ as defined in Def. 1.1.12 of our problem setting in Section 1.2.2

#### Adapting neighborhood measures

Neighborhood measures, defined in Eq. 8.1, 8.3, 8.4 and 8.5, take the following form in drug-target formulations, for common neighbors (CN), the Jaccard coefficient (JC), preferential

attachment (PA), and SimRank (SR) respectively:

$$CN(d_i,t_j) = |\{v \mid (d_i,v) \in E\} \cap \{u \mid (t_j,u) \in E\}|, \tag{8.11}$$

$$JC(d_i,t_j) = \frac{CN(d_i,t_j)}{|\{v \mid (d_i,v) \in E\} \cup \{u \mid (t_j,u) \in E\}|}, \tag{8.12}$$

$$PA(d_i,t_j) = ||\Gamma(d_i)|| \cdot ||\Gamma(t_j)||, \text{ with } ||\Gamma(d_i)|| = \sum_{k=1}^{|V|} A_{ik}, ||\Gamma(t_j)|| = \sum_{k=1}^{|V|} A_{jk}, \tag{8.13}$$

$$SR(d_i,t_j) = \frac{CN(d_i,t_j)}{PA(d_i,t_j)}. \tag{8.14}$$

$||\Gamma(v)||$ denotes the weight of a neighborhood of a vertex $v$, which for individual vertices is equivalent to their *degree* (defined in Def. 1.1.16 in Chapter 1). Note, that operator $||\Gamma(v)||$ defined in Eq. 8.13 is an extension of $|\Gamma(v)|$ in Eq. 8.4 to weighted graphs, the case of our problem setting.

Due to the nature of the communities we obtain, there is little overlap between vertices' neighborhoods, preventing the direct use of neighborhood-based measures. To overcome this, we adapt neighborhood measures for use with our communities, treating them like neighborhoods:

1. The CN measure turns the number of common neighbors of communities $d_i$ and $t_j$ into the number of connections. Instead of the measure from Eq. 8.11, we define the $\text{CN}_{CC}$ version corresponding to *CC* matching:

   $$CN_{CC}(d_i,t_j) = |\{(d,t) \in E \mid d \in C(d_i), t \in C(t_j)\}|, \tag{8.15}$$

   The *NC* version of CN is defined as the average of the two independent predictions, $\text{CN}_{NC}(d_i)$ and $\text{CN}_{NC}(t_j)$, for $d_i$ and $t_j$ respectively:

   $$CN_{NC}(d_i,t_j) = \frac{1}{2}\left(CN_{NC}(d_i) + CN_{NC}(t_j)\right), \tag{8.16}$$

   with $\text{CN}_{NC}(d_i)$ and $\text{CN}_{NC}(t_j)$ defined as:

   $$CN_{NC}(d_i) = |\{t \mid (d_i,t) \in E, t \in C(t_j)\}|,$$

   $$CN_{NC}(t_j) = |\{d \mid (t_j,d) \in E, d \in C(d_i)\}|.$$

2. JC represents the fraction of all possible connections of $d_i$ and $t_j$ that are connected to both. Using the *CC* and *NC* formulations, our bipartite adaptations of Eq. 8.12 take the form:

   $$JC_{CC}(d_i,t_j) = \frac{CN_{CC}(d_i,t_j)}{|C(d_i)| \cdot |C(t_j)|}, \tag{8.17}$$

   $$JC_{NC}(d_i,t_j) = \frac{1}{2}\left(\frac{CN_{NC}(d_i)}{|C(t_j)|} + \frac{CN_{NC}(t_j)}{|C(d_i)|}\right), \tag{8.18}$$

   where $C(d_i)$, $C(t_j)$ denote sets of nodes that the communities of a drug $d_i$ and a target $t_j$ have, respectively, and $|C(d_i)|$, $|C(t_j)|$ represent their sizes.

3. PA is defined as the product of degrees of communities of $d_i$ and $t_j$. The *CC* formulation of the measure from Eq. 8.13 takes the form:

$$PA_{CC}(d_i,t_j) = ||\Gamma(C(d_i))|| \cdot ||\Gamma(C(t_j))||, \tag{8.19}$$

where $||\Gamma(C(v))||$ denotes the degree of the neighborhood of a community, $v$ defined as:

$$||\Gamma(C(v))|| = \sum_{v \in C(v), u \notin C(v)} A(v,u), \tag{8.20}$$

with the neighborhood of a community defined as the set of neighbor vertices the community has: $\Gamma(C(v)) = \{v \mid v \in C(v), u \notin C(v), (u,v) \in E\}$.
Taking into account that $\text{PA}_{NC}(d_i) = ||\Gamma(d_i)|| \cdot ||\Gamma(C(t_j))||$ and $\text{PA}_{NC}(t_j) = ||\Gamma(t_j)|| \cdot ||\Gamma(C(d_i))||$ we can define the *NC* version for PA (Eq. 8.13):

$$PA_{NC}(d_i,t_j) = \frac{1}{2}(PA_{NC}(d_i) + PA_{NC}(t_j)). \tag{8.21}$$

4. Finally, SR is equal to the number of connections of the communities of $d_i$ and $t_j$ normalized by the product of their degrees. Instead of the measure from Eq. 8.14, we define $\text{SR}_{CC}$ and $\text{SR}_{NC}$ versions corresponding to *CC* and *NC* matching respectively:

$$SR_{CC}(d_i,t_j) = \frac{CN_{CC}(d_i,t_j)}{PA_{CC}(d_i,t_j)}, \tag{8.22}$$

$$SR_{NC}(d_i,t_j) = \frac{1}{2}\left(\frac{CN_{NC}(d_i)}{PA_{NC}(d_i)} + \frac{CN_{NC}(t_j)}{PA_{NC}(t_j)}\right). \tag{8.23}$$

#### Adapting community-based measures

Community-based measures are based on one or several of the following assumptions: all vertices have the same semantic, all edges have the same semantic, edges are unweighted, or vertices whose link is to be predicted find themselves in the same community. We therefore cannot use them in a straightforward manner, but we can adapt them to our bipartite setting:

1. CAR-based measures exploit the density of communities to reward (or penalize) densely (sparsely) connected neighbors of the vertices whose link is to be predicted. In other words, they are based on the assumption that common neighbors of the given vertices should belong to the same community, in which case the probability of the connection will be highest. We thus cannot adapt the measures to the *CC* formulation, which assumes that the vertex communities are separated, but can propose the *NC* versions.
Our adapted *CAR-based common neighbors* (CCN) will be defined as CN regularized by local community degree, in turn defined as the sum of weights of all edges inside a community. Using Eq. 8.6, 8.16, the *NC* formulation of CCN can be derived as:

$$CCN_{NC}(d_i,t_j) = \frac{1}{2}\left(CCN_{NC}(d_i) + CCN_{NC}(t_j)\right), \tag{8.24}$$

with $\text{CCN}_{NC}(d_i)$ and $\text{CCN}_{NC}(t_j)$ in turn defined as:

$$CCN_{NC}(d_i) = |\{t \mid (d_i,t) \in E, t \in C(t_j)\}| \cdot \sum_{t_l,t_k \in C(t_j)} A(t_l,t_k) \text{ and}$$

$$CCN_{NC}(t_j) = |\{t \mid (t_j,d) \in E, d \in C(d_i)\}| \cdot \sum_{d_l,d_k \in C(d_i)} A(d_l,d_k).$$

In the same manner, using Eq. 8.7, 8.18, we redefine the *CAR-based Jaccard coefficient* (CJC) as CCN normalized by the size of the community:

$$CJC_{NC}(d_i,t_j) = \frac{1}{2}\left(CJC_{NC}(d_i) + CJC_{NC}(t_j)\right), \tag{8.25}$$

$$\text{with } CJC_{NC}(d_i) = \frac{CCN_{NC}(d_i)}{|C(t_j)|} \text{ and}$$

$$CJC_{NC}(t_j) = \frac{CCN_{NC}(t_j)}{|C(d_i)|}.$$

2. The neighboring community-based measure estimates the probability that two given nodes belong to the same community. As in the case of CAR-based measures, we cannot define a *CC* formulation because of the same reason, and thus propose only an *NC* formulation.
Using Eq. 8.8, we define the *Neighboring community-based* (NCB) measure as the normalized sum of all CN regularized by the size of the respective community:

$$NCB_{NC}(d_i,t_j) = \frac{1}{2}\left(NCB_{NC}(d_i) + NCB_{NC}(t_j)\right), \tag{8.26}$$

with $\text{NCB}_{NC}(d_i)$ and $\text{NCB}_{NC}(t_j)$ in turn:

$$NCB_{NC}(d_i) = \sum_{k=1}^{c_t} \frac{|\{t \mid (d_i,t) \in E, t \in C_k\}|}{|C_k|} \cdot \frac{|\{t \mid t \in C_k\}|}{|T|} \text{ and}$$

$$NCB_{NC}(t_j) = \sum_{k=1}^{c_d} \frac{|\{d \mid (t_j,d) \in E, d \in C_k\}|}{|C_k|} \cdot \frac{|\{d \mid d \in C_k\}|}{|D|}.$$

Moreover, assuming communities are pure, i.e. consisting only of either drugs or targets, these equations can be simplified to the sum of all CN normalized by the number of vertices of one type:

$$NCB_{NC}(d_i) = \frac{1}{|T|}\sum_{k=1}^{c_t} |\{t \mid (d_i,t) \in E, t \in C_k\}|,$$

$$NCB_{NC}(t_j) = \frac{1}{|D|}\sum_{k=1}^{c_d} |\{d \mid (t_j,d) \in E, d \in C_k\}|.$$

3. The community relevance measures extend the notion of node neighborhoods in neighborhood measures to neighborhoods of communities.
Using Eq. 8.9, we define the *CC* version of the JC measure as the number of common nodes of examined communities and nodes of the opposite type connected to those communities, normalized by the total number of nodes in this selection. We refer to it as *Community relevance Jaccard coefficient* (CRJC):

$$CRJC_{CC}(d_i,t_j) = \frac{|CRJC_{CC}(d_i) \cap CRJC_{CC}(t_j)|}{|CRJC_{CC}(d_i) \cup CRJC_{CC}(t_j)|}, \tag{8.27}$$

with $\text{CRJC}_{CC}(d_i)$ and $\text{CRJC}_{CC}(t_j)$ in turn:

$$CRJC_{CC}(d_i) = \{t \mid (d,t) \in E, d \in C(d_i)\} \cup \{d \mid d \in C(d_i)\} \text{ and}$$

$$CRJC_{CC}(t_j) = \{d \mid (t,d) \in E, t \in C(t_j)\} \cup \{t \mid t \in C(t_j)\}.$$

Since the adapted versions of Community relevance are not based on counting specific edges (as in the case of adapted neighborhood measures), but on counting nodes instead,

the *NC* version of CRJC for a single node would be the same as $\text{CRJC}_{CC}$ for the same node in our interpretation. In other words, there is no sense to define a separate *NC* version of the measure. For the same reason, Community relevance versions of other neighborhood measures become meaningless too: just counting intersecting nodes instead of edges in case of CN, multiplying numbers of neighbors of communities of the two nodes in case of PA, and the same for their combination in case of SR. This is why we define the JC version of Community relevance only.

### 8.2.3 Auto-tuning of parameters

In link prediction, as in any predictive task, an important issue is choosing optimal parameter values. In our approach these parameters are limited to the ones of the community detection algorithm that is used.

The main constraint of spectral partitioning is that one parameter, number of eigen values $m$, has to be preselected by the user before running the approach, because there is no recommended nor default number for it. The thresholding function, on the other hand, has a default value, but trying other functions can help to improve the quality of the prediction. Louvain does not have a parameter that has to be preselected by the user, however optimizing its only parameter, resolution limit, can help a bit to improve prediction as well.

In the absence of other knowledge, one would use *cross-validation* as a systematic method to optimize parameters. Several-fold cross-validation is also the method of choice to evaluate the performance of a classifier, however, so that we propose to use a double cross-validation: splitting off an *external* test fold of present edges to evaluate the method, and using an *internal* cross-validation to optimize parameters of the community detection algorithm used.

We will use this method in Chapter 9, in which we perform evaluation of our LPBYCD approach, to auto-tune the approach parameters.

### 8.2.4 Example

In order to illustrate how our method works in practice, let us consider an example on how two of the most common link prediction measures, common neighbors and the Jaccard coefficient, are computed using both matching techniques, *CC* and *NC*. Given a network as on figure Fig. 8.9 which has 5 drugs, $l1,\ldots,l5$, grouped into 2 communities $C1$ and $C2$, and 4 targets, $t1,\ldots,t4$, grouped into 2 communities $C3$ and $C4$, we want to estimate the probability of links between drug $l2$ and target $t2$, as well as between drug $l5$ and target $t3$.

#### Community to community matching

**Common neighbors** In *CC* matching, common neighbors can be easily computed using Eq. 8.15: $l2$ belongs to community $C1$, $t2$ belongs to community $C3$, and there are 3 links between communities $C1$ and $C3$. Analogously, $l5$ belongs to $C2$, $t3$ is part of $C4$ and only one link exists between these two communities (Fig. 8.10):

$$CN_{CC}(l2,t2) = 3,$$

$$CN_{CC}(l5,t3) = 1.$$

**Jaccard coefficient** The Jaccard coefficient can be computed as the fraction of existing links between node communities and the total possible number of links between them. The number of existing links equals the common neighbors measure, and the possible number of links between communities can be computed as the product of their sizes: for the $l2$-$t2$ pair it equals 6, because

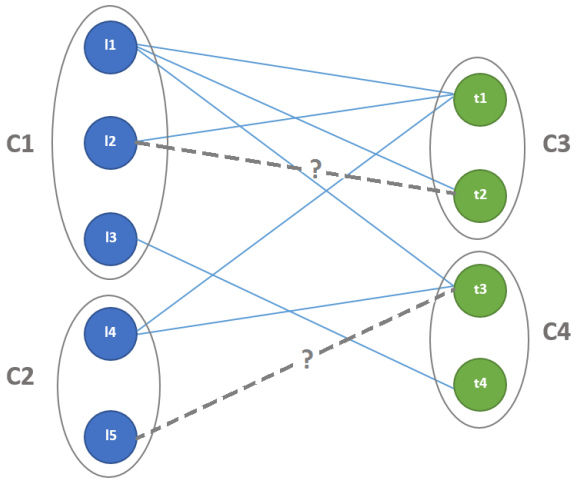


Figure 8.9: An example network with 7 existing edges (in blue) and 2 examined connections (in dashed grey)

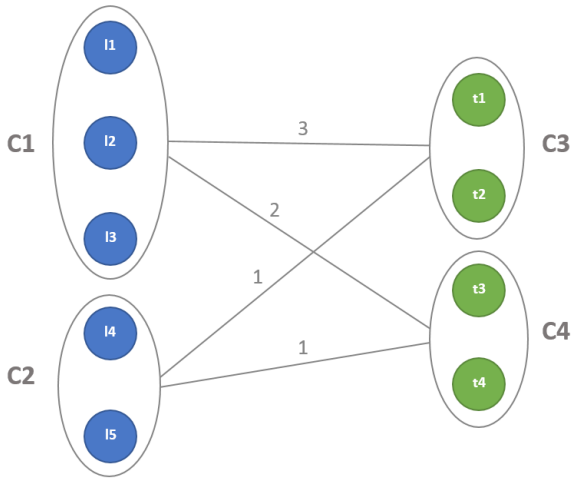


Figure 8.10: CC matching on example network from Fig. 8.9 (label on edges represent number of existing edges between vertices of matched communities)

there are 3 drugs in community $C1$ and 2 targets in $C3$, and for $l5$-$t3$ it is 4, because $C2$ and $C4$ have 2 nodes each (Fig. 8.10). Now the JC can be computed using Eq. 8.17:

$$JC_{CC}(l2,t2) = \frac{3}{6} = 0.5,$$

$$JC_{CC}(l5,t3) = \frac{1}{4} = 0.25.$$

### Node to community matching

**Common neighbors** In *NC* matching, we first match drug nodes with target communities, then drug communities with targets, and perform prediction twice (Eq. 8.16). For the first matching, there only one link exists between drug $l2$ and community $C3$, and for the second, there is no link between $l5$ and $C4$ (Fig. 8.11):

$$CN_{NC1}(l2,t2) = 1,$$

$$CN_{NC1}(l5,t3) = 0.$$

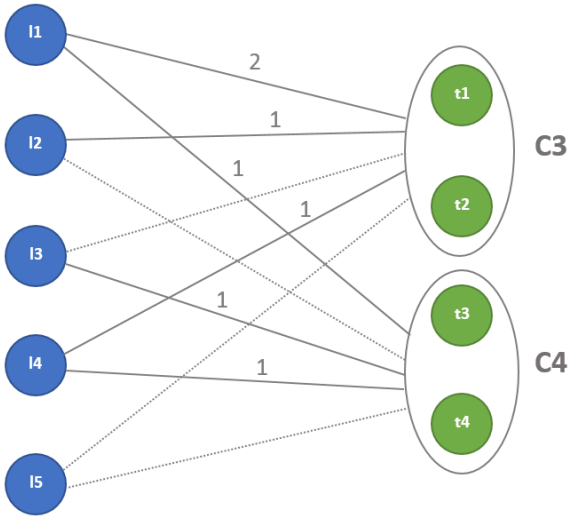


Figure 8.11: Drug nodes to target community matching on example network from Fig. 8.9 (label on edges represent the number of existing edges between drug nodes and vertices of matched target communities, and dotted edges represent non-interacting pairs)

For the second matching, targets to drug communities, common neighbors of the first pair is 1, because only one link exists between $C1$ and $t2$, and for the second pair it also equals 1 as $C2$ and $t3$ also share only one connection (Fig. 8.12):

$$CN_{NC2}(l2,t2) = 1,$$

$$CN_{NC2}(l5,t3) = 1.$$

Now, an average of the 2 independent predictions we will result in the following:

$$CN_{NC}(l2,t2) = \frac{CN_{NC1}(l2,t2)+CN_{NC2}(l2,t2)}{2} = \frac{1+1}{2} = 1,$$

$$CN_{NC}(l5,t3) = \frac{CN_{NC1}(l5,t3)+CN_{NC2}(l5,t3)}{2} = \frac{0+1}{2} = 0.5.$$

**Jaccard coefficient** In the first matching, drugs to target communities, $\mathrm{CN}_{NC1}(l2,t2) = 1$ for the first pair and there are 2 links possible between $l2$ and $C3$. For the second pair, $\mathrm{CN}_{NC1}(l5,t3) = 0$ and there are also only 2 links possible between $l5$ and $C4$ (Fig. 8.11). According to Eq. 8.18, the JC will be equal to:

$$JC_{NC1}(l2,t2) = \frac{1}{2} = 0.5,$$

$$JC_{NC1}(l5,t3) = \frac{0}{2} = 0.$$

For the second matching, only one link exists between $C1$ and $t2$ out of 3 possible and $\mathrm{CN}_{NC2}(l2,t2) = 1$ for the first pair, and only one link exists between $C2$ and $t3$ out of 2, and $\mathrm{CN}_{NC2}(l5,t3) = 1$ for the second (Fig. 8.12):

$$JC_{NC2}(l2,t2) = \frac{1}{3} = 0.33,$$

$$JC_{NC2}(l5,t3) = \frac{1}{2} = 0.5.$$

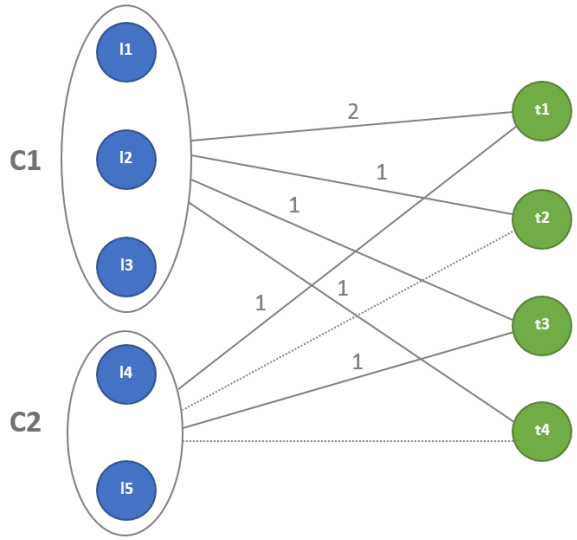


Figure 8.12: Drug communities to target nodes matching on example network from Fig. 8.9 (label on edges represent number of existing edges between vertices of drug communities and matched target nodes, and dotted edges represent non-interacting pairing)

Taking an average between two predictions results in the following:

$$JC_{NC}(l2,t2) = \frac{JC_{NC1}(l2,t2) + JC_{NC2}(l2,t2)}{2} = \frac{0.5+0.33}{2} = 0.42,$$

$$JC_{NC}(l5,t3) = \frac{JC_{NC1}(l5,t3) + JC_{NC2}(l5,t3)}{2} = \frac{0+0.5}{2} = 0.25.$$

**Discussion**

In the given example we estimated the probability of interaction for 2 drug-target pairs $l2$ and $t2$, and $l5$ and $t3$ using 2 different matching and 2 different methods. Both measures in both matchings showed that the first pair is more probable than the second one. However, according to the *CC* version of common neighbors the first link is three times as probable as the second (3 vs. 1, for the first vs. second pair respectively), while using the *NC* version this difference is only twice (1 vs. 0.5). In the case of the Jaccard coefficient this difference shrinks, and the gap between different formulations too. According to $JC_{CC}$ the first link is twice probable than the second (0.5 vs. 0.25), while with $JC_{NC}$ is only 1.68 (0.42 vs. 0.25). The Jaccard coefficient has more cautious behavior. That was expected, because *JC* is the normalized measure. And our example demonstrates that this behavior is preserved after the measures had been adapted to our setting.

## 8.3 Conclusion

We developed an approach for link prediction using network's community information. The proposed example illustrates how our method can be applied to predict new interactions in a network with bipartite origin. The LPBYCD approach outputs numerical values corresponding to different measures as the result of an interaction prediction. However, our problem setting defined in Section 1.2.2 requires a binary decision to be made on the prediction result. To overcome it, the predictions can be ranked according to different measures and a selected number of top ones can be presented as the final result, for instance using a predefined threshold and best performing measures. In the next chapter, we will extend this idea by performing an evaluation on real data and demonstrating performance of our approach on that.

# 9. Experimental evaluation of predicting links with communities

*In this chapter, I present and discuss the experimental results of the* LPBYCD *approach developed in Chapter 8 to solve the link prediction problem defined in Chapter 1. I start with a description of the experimental setup. I then present results with different settings using different data sets assessing the general performance of the approach as well as its scalability, genericness and interpretability. Finally, I perform validation of link prediction founded on external knowledge using the framework described in Chapter 3. I then finish the chapter with the conclusion.*

## 9.1 Experimental setup

In this section, I describe the parameters of the approach and the experimental protocol. The latter includes the description of the data sets, the evaluation protocol, the quality measures and implementation details.

### 9.1.1 Parameter selection

#### Community detection methods

We test the LPBYCD approach with two community detection methods, spectral partitioning and the Louvain algorithm, presented in Section 8.1.3.

We apply spectral partitioning to multi-layer graphs by "flattening" the graph, i.e. summing edge weights to derive the adjacency and degree matrices before performing partitioning. Since Louvain does not employ matrices, we translate the graph into a single-layer graph by summing up the weights of all edges between two vertices. Practically, the result of the last is the same as of "flattening" the graph for spectral partitioning, but technically it is performed differently.

#### Parameters to optimize

As described in Section 8.1.3, spectral partitioning has two parameters: the value of $m$ and the thresholding method. The $m$ parameter partitions the graph into $2^m$ groups at most, and the thresholding method affects the quality of the partitioning.

As thresholding function we use *sign cut* (which we refer to as *default*), *bisection cut* (or *median*) and *mean*, which were presented in Section 8.1.3. In addition to them we propose *sum*, which exploits the fact that there are approximately equally as many positive and negative values in eigenvectors of Laplacian matrix, i.e. in practice they sum to a value close to zero. This relatively

small threshold value satisfies the first requirement of the partitioning task [149], creating groups of vertices approximately equal in size.

Moreover, the same thresholding function can be applied to all eigenvectors, or each individual eigenvector can have its own threshold. We call the former approach *global*, and the latter *individualized*. Additionally, the global threshold can be computed by applying the aggregating function (mean, median or sum) to all eigenvectors or only to the $m$ actually used. We refer to this latter type as *localized*. To sum up, we evaluate nine different thresholding methods: global, localized, individualized and their combinations with mean, median and sum. The combination *global sum* is a special case since taking the first eigenvector, whose entries all have the same, positive value, into account violates the "close to zero" property sum thresholding exploits. We therefore do not evaluate that thresholding method, but add *default* thresholding to the mix for the experimental evaluation.

The Louvain algorithm has only one parameter – *resolution limit*. It defines a modularity scale which was explained in detail in Section 8.1.3. The resolution limit is not required to be preselected by the user as the $m$ value is in spectral partitioning. Instead, the default value usually gives satisfying results. However, as in the case of selecting the optimal thresholding method in spectral partitioning, the resolution limit value can be optimized to get a better result.

### 9.1.2 Experimental protocol

#### Data sets

We perform our experiments on the benchmark data sets presented in Section 2.1.2: Enzyme, GPCR, IC, NR and Kinase. The data consist of three networks: drug similarities, target similarities and drug-target interactions (the bipartite graph). The data sets' basic properties are presented in Table 2.2.

In addition, we evaluate our approach on a bigger data set, IUPHAR network, introduced in Section 2.1.1 and used before to evaluate NEWERMINE in Chapter 6. IUPHAR data consists of 6 layers: drug similarity, drug interaction from the literature, 2 target similarity networks, target interaction from the literature and the bipartite layer with drug-target interactions. The IUPHAR basic properties are presented in the last row of Table 2.1.

Finally, we evaluate the generic side of our approach by performing experiments on data that comes from a different domain than drug-target interactions. These are 2 generic data sets: MovieLens and Unicode languages introduced in Section 2.2. Each of these sets is represented by three networks: the bipartite graph plus similarity layers of each vertex type. Their basic properties are presented in Table 2.4.

#### Evaluation protocol

To perform evaluation of our experiments, we used a *5×5-fold cross-validation*, with each fold containing 20% of all edges implemented as follows. In each round the test set contains one fifth of all drug-target interactions, acting as test set for link prediction once, while community detection is performed on the other 80%. The process is repeated 5 times, the results are averaged among all runs.

#### Quality measures

We evaluate all the predictions by AUC and AUPR (Section 1.1.3), averaging the results. We also report standard deviation values (std) when it is applicable, to demonstrate the stability of our approach.

#### Implementation

I implemented spectral partitioning with all thresholding methods and all link prediction measures in Python. I made use of the following libraries for that: the networkx library to model

the multi-layer network, NumPy for all matrix computations and sklearn to calculate the curves. As for Louvain, the python-louvain package was used as the algorithm implementation.

## 9.2 Experimental results

We begin by evaluating the different measures described in Section 8.2.2 with both community detection approaches, spectral partitioning and the Louvain algorithm, and select the best performing measure. Following this, we use this measure to optimize the community detection parameters (the threshold type and *m* for spectral partitioning, and resolution limit for the Louvain algorithm) to gain an idea of the best attainable performance, which we compare with NEWERMINE and state of the art methods. At the same time, we demonstrate that those parameters can be set internally using an approach described in Section 8.2.3. We also perform experiments on the larger and more challenging IUPHAR data set and test the scalability of our approach. In addition to that we evaluate genericness by doing experiments on the data sets described in Section 2.2. We finish by explaining the potential interpretability of the results our approach can provide.

### 9.2.1 Link prediction measures evaluation

We first test the different link prediction measures from Section 8.2.2 on communities produced by either spectral partitioning or the Louvain algorithm. To reduce computational complexity, we use default parameters for community detection: *default* as threshold for spectral partitioning and 1.0 as resolution limit for the Louvain algorithm. Note that we optimize *m* for spectral partitioning on the test data in this experiment, because, as it was mentioned in Section 9.1.1, there is no a priori number of eigenvalues that will fit all data. The results are presented in Fig. 9.1, with *CC* formulations on the left, *NC* ones on the right of each plot. The best-performing measure for each group is indicated by a + sign over the corresponding bar. We also report on the figure the optimal *m* value for every measure in spectral partitioning.

The results show that a number of measures, e.g. $SR_{CC}$, $CN_{NC}$, $SR_{NC}$, $CCN_{NC}$, $CJC_{NC}$, $NCB_{NC}$, have acceptable performance in terms of AUC on most of data sets while the Jaccard coefficient performs best for both the *CC* and *NC* versions in Enzyme, GPCR, IC and NR sets. It might happen because the JC gives less extreme values due to normalization (as was shown in Section 8.2.4), also it takes into account sizes of communities, while most of the other measures (CN, PA, SR, CCN and NCB) do not. For Kinase, this is only partially true ($NCB_{NC}$ is the best in spectral partitioning and $CN_{NC}$ performs the same as $JC_{NC}$ for Louvain), which can be explained by the fact that Kinase is a quite specific set (has much more targets than drugs, not as sparse as others), and comes from a different source (see Section 2.1.2). Concerning comparison in terms of AUPR, the *NC* version of the JC performs the best (with only the exception of spectral partitioning on Kinase where it is very close to the maximum). As for the *CC* version, this is only true when using Louvain: the JC is the best (as in case of IC) or very close to the maximum (as in case of other sets). The exception for this is spectral partitioning where the JC is only second best to be outperformed by PA and SR measures alternately. *We will therefore use $JC_{CC}$ and $JC_{NC}$ as best performing measures in the following experiments, and test the* LPBYCD *approach further with more diverse data.*

Another result is that for these parameter settings spectral partitioning and the Louvain algorithm give approximately the same AUC (blue bars in Fig. 9.1), but the latter improves on AUPR (red bars in the figure). *This is not enough to draw a conclusion that Louvain is always better, and we will continue testing both community detection approaches going forward.*

Finally, using *NC* predictions requires a lower *m*, i.e. less fine-grained partitions, for spectral partitioning (the majority of optimal m values are less than 10 on the right of each plot, while it is the opposite for most *CC* predictions on the left). *NC* is a more expensive approach than *CC* in terms of computational time (requires more operations on matching and performs predictions


Figure 9.1: Link prediction measures evaluation on the benchmark data sets. The symbol + denotes the best performing measure for each group of formulations in each data set

Table 9.1: Spectral partitioning parameter optimization via internal cross-validation

| Data set | Measure | Internal validation | | | | External validation | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | *std* | AUPR | *std* | AUC | *std* | AUPR | *std* |
| Enzyme | $JC_{CC}$ | 0.85 | 0.00 | 0.14 | 0.04 | 0.85 | 0.01 | 0.19 | 0.07 |
| | $JC_{NC}$ | 0.91 | 0.00 | 0.19 | 0.02 | 0.92 | 0.01 | 0.26 | 0.06 |
| GPCR | $JC_{CC}$ | 0.79 | 0.01 | 0.11 | 0.02 | 0.80 | 0.02 | 0.15 | 0.02 |
| | $JC_{NC}$ | 0.84 | 0.01 | 0.19 | 0.01 | 0.85 | 0.01 | 0.25 | 0.02 |
| IC | $JC_{CC}$ | 0.82 | 0.01 | 0.31 | 0.03 | 0.83 | 0.02 | 0.40 | 0.04 |
| | $JC_{NC}$ | 0.87 | 0.00 | 0.32 | 0.06 | 0.88 | 0.01 | 0.41 | 0.05 |
| NR | $JC_{CC}$ | 0.73 | 0.02 | 0.21 | 0.03 | 0.72 | 0.06 | 0.24 | 0.08 |
| | $JC_{NC}$ | 0.75 | 0.03 | 0.19 | 0.02 | 0.77 | 0.06 | 0.23 | 0.13 |
| Kinase | $JC_{CC}$ | 0.76 | 0.01 | 0.17 | 0.01 | 0.77 | 0.03 | 0.23 | 0.02 |
| | $JC_{NC}$ | 0.85 | 0.00 | 0.26 | 0.02 | 0.86 | 0.01 | 0.35 | 0.02 |

twice), thus this requirement compensates the final cost when searching for the optimal $m$.

### 9.2.2 Parameter selection via internal cross-validation

As presented in Section 8.2.3, we propose to use *internal cross-validation* (internal CV) as a systematic way to fix community detection approach parameters. We implement internal CV as follows: splitting off each of 5 *external* test folds (containing 20% of present edges each) to evaluate the model, and using an *internal* cross-validation on the remaining 80% training data as described in Section 9.1.2 to fix the model's parameters. During the internal CV, we perform grid search over the different parameter settings, varying $m$ in the interval $[1,25]$, and testing this value with all nine options for thresholding for spectral partitioning, and varying resolution in the interval $[0.1,1]$ using steps of 0.1 for Louvain. We report averaged results using mean and std.

Table 9.1 presents the results for spectral partitioning. It shows both the results of internal evaluation, i.e. on the validation set used to fix parameter values, and of the external evaluation, i.e. on the unseen training data. The results between internal and external validation align very closely w.r.t. AUC (the difference between internal and external AUC values is 0.02 at most). These values are similar even on individual folds (see Table 1–3 in Appendix 1 for exact values). At the same time, relatively low std values show that AUC/AUPR values for different folds do not differ much. These observations allow us to draw the conclusion that *the results are rather stable and there is no risk of overfitting when building the model*. Concerning AUPR, interestingly enough, the results on the testing folds are in fact *higher* than for the validation data used in the internal validation. For Enzyme and NR, this might be explainable by slightly higher std values, i.e. for some folds the result remains approximately the same, for others it is an improvement. In addition, *using NC matching always gives better results than CC matching, often by a large margin*.

As in the case of spectral partitioning, the performance of Louvain summarized in Table 9.2 is rather close in terms of AUC (the difference between internal and external AUC values is 0.03 at most for all sets except of GPCR), but the differences in AUPR are even more pronounced (the differences are more than 0.1 or ten percentage points). GPCR is a bit of an outlier in this comparison, since the differences between internal quality estimation and test fold results are larger than for the other sets (0.73 and 0.85 vs. 0.8 and 0.89 for $JC_{CC}$ and $JC_{NC}$ measures respectively). *This motivates us to also test our approach with more diverse data*. Concerning the superior performance of *NC* matching, as in the case of spectral partitioning this also holds for Louvain.

More detailed results w.r.t. each fold, shown in Table 1–3 in Appendix 1 and Table 4, 5 in Appendix 1 for spectral partitioning and Louvain respectively, demonstrate that it is rather important to optimize approach parameters on different data, represented by different folds in these

Table 9.2: Louvain algorithm resolution optimization via internal cross-validation

| Data set | Measure | Internal validation | | | | External validation | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | *std* | AUPR | *std* | AUC | *std* | AUPR | *std* |
| Enzyme | $JC_{CC}$ | 0.94 | 0.00 | 0.52 | 0.00 | 0.96 | 0.00 | 0.69 | 0.04 |
| | $JC_{NC}$ | 0.95 | 0.00 | 0.56 | 0.02 | 0.96 | 0.01 | 0.71 | 0.02 |
| GPCR | $JC_{CC}$ | 0.73 | 0.01 | 0.18 | 0.02 | 0.80 | 0.03 | 0.31 | 0.05 |
| | $JC_{NC}$ | 0.85 | 0.01 | 0.28 | 0.03 | 0.89 | 0.01 | 0.49 | 0.04 |
| IC | $JC_{CC}$ | 0.94 | 0.00 | 0.50 | 0.01 | 0.95 | 0.01 | 0.65 | 0.04 |
| | $JC_{NC}$ | 0.96 | 0.00 | 0.61 | 0.02 | 0.97 | 0.01 | 0.78 | 0.03 |
| NR | $JC_{CC}$ | 0.79 | 0.03 | 0.27 | 0.06 | 0.82 | 0.06 | 0.45 | 0.13 |
| | $JC_{NC}$ | 0.81 | 0.03 | 0.30 | 0.05 | 0.80 | 0.09 | 0.42 | 0.14 |
| Kinase | $JC_{CC}$ | 0.72 | 0.00 | 0.15 | 0.01 | 0.73 | 0.01 | 0.19 | 0.01 |
| | $JC_{NC}$ | 0.90 | 0.00 | 0.38 | 0.01 | 0.91 | 0.01 | 0.50 | 0.03 |

experiments.

### 9.2.3 Comparison with a random walk method

We also compare the results with the ones of NEWERMINE, our random walk approach presented in Section 5.2. We run NEWERMINE with the recommended parameter settings from Section 6.2 ($\eta = 0.2$, $\beta = 0.7$), and we use 5x5 cross-validation to make a fair comparison (Table 9.3).

Table 9.3: The LPBYCD result comparison with a random walk method

| Data set | NEWERMINE | | Spectral part. | | Louvain | |
|---|---|---|---|---|---|---|
| | AUC | AUPR | AUC | AUPR | AUC | AUPR |
| Enzyme | 0.84 | 0.15 | 0.92 | 0.26 | 0.96 | 0.71 |
| GPCR | 0.80 | 0.22 | 0.85 | 0.25 | 0.89 | 0.49 |
| IC | 0.76 | 0.27 | 0.88 | 0.41 | 0.97 | 0.78 |
| NR | 0.63 | 0.26 | 0.77 | 0.24 | 0.82 | 0.45 |
| Kinase | 0.61 | 0.13 | 0.86 | 0.35 | 0.91 | 0.45 |

As a result, *$JC_{CC}$ and $JC_{NC}$ in combination with both spectral partitioning and Louvain clearly outperform* NEWERMINE *in terms of both AUC and AUPR*. The only exception is AUPR of NR for spectral partitioning, which could be due to the relatively small size of the set for a spectral method to detect proper communities.

### 9.2.4 Comparison to the state of the art

Concerning comparison with the state of the art, *the* LPBYCD *approach initialized by the Louvain algorithm and $JC_{NC}$ measure comes close to the performance of the state of the art* methods reported in [28] in terms of AUC:

- Enzyme: 0.97 compared to 0.96 for our method with Louvain,
- GPCR: 0.95 compared to 0.89,
- IC: 0.98 compared to 0.97,
- NR: 0.88 compared to 0.8, and
- Kinase: 0.93 compared to 0.91, respectively.

For time saving reasons, we do not reconstruct these experiments ourselves but use the reported values in the paper.

Table 9.4: Performance ceiling on IUPHAR with optimal parameters

| Method | Optimal parameters | Performance | | | | Running time[a], s |
|---|---|---|---|---|---|---|
| | | AUC | *std* | AUPR | *std* | |
| NEWERMINE | $\eta = 0.2, \beta = 0.7$ | 0.57 | - | - | - | $11.9 \times 8137$[b] |
| Spectral part. ($JC_{CC}$) | m=400, default[c] | 0.74 | 0.02 | 0.01 | 0.00 | 3477.82 |
| Spectral part. ($JC_{NC}$) | m=0[d] | 0.85 | 0.00 | 0.01 | 0.00 | 2484.91 |
| Louvain ($JC_{CC}$) | resolution=0.1 | 0.61 | 0.01 | 0.10 | 0.01 | 5263.97 |
| Louvain ($JC_{NC}$) | resolution=0.8 | 0.82 | 0.01 | 0.02 | 0.00 | 2702.62 |

[a] for 1 fold in average
[b] must be run for each drug
[c] without thresholding optimization
[d] without threshold

### 9.2.5 Performance on a bigger data set

The five benchmark data sets on which we used up to now are relatively small and dense, as shown in Table 2.2. In this section, we therefore contrast those results with ones using the IUPHAR network with 6 layers. The data set basic properties presented in Table 2.1 show that IUPHAR is significantly larger. This is the same set that was used to evaluate NEWERMINE in Chapter 6. Note, that we do not imitate benchmark data set structure by removing negative edges from the IUPHAR. Instead, we ignore edge labels for measures computation step, what allows us to make a comparison of the results with NEWERMINE. As before, we perform evaluation using spectral partitning and the Louvain algorithm with the Jaccard coefficient (as best performing measure) in both formulations, *CC* and *NC*.

Given that we have shown in Section 9.2.2 above that fixing parameter values internally gives effectively the same results as reporting the best result on the testing data, we did the latter for IUPHAR to save time. The results are presented in Table 9.4. NEWERMINE results are taken from Chapter 6, where AUPR was not used as quality measure, and thus was not computed. Concerning spectral partitioning, IUPHAR is too large to search the parameter space for $m$ starting from $m = 1$ with step size 1 – instead we used step size 10 for the grid search and searched in a more fine-grained manner once we had found a maximum in this way. As for Louvain, resolution was optimized in the interval $[0.1, 1]$ using steps of 0.1 as before.

As Table 9.4 shows, *using either spectral partitioning or Louvain clearly improves on the results of* NEWERMINE. Contrary to the results on the benchmark data sets, however, spectral partitioning *outperforms* Louvain, even though the two are close for *NC* matching. The table also shows a very surprising result in that spectral partitioning with *NC* version of the Jaccard coefficient does best when not creating communities *at all*. In that case it is the matching technique that does the heavy lifting and effectively matches each drug with the community containing *all* targets and each target with the community containing *all* drugs.

To evaluate whether this is a phenomenon that is specific to IUPHAR or occurs more generally, we compare the results for $m = 0$ on the benchmark data to the ones achieved by parameter-optimized spectral partitioning in Table 9.5. As the table shows, optimizing parameters *does* provide a performance gain, sometimes strongly so, as in the case of NR. Yet at the same time, the results for $m = 0$ are acceptable. This indicates both that those *benchmark data sets are not fully representative of the problem setting*, and that *for large data one could use a quick-shot prediction* with $m = 0$ and *NC* matching before trying to fully optimize parameters.

Table 9.5: Link prediction without community detection vs spectral partitioning with optimal parameters and *NC* matching

| Data set | Without communities (m=0) | | | | Optimal m and threshold | | | |
|---|---|---|---|---|---|---|---|---|
| | AUC | *std* | AUPR | *std* | AUC | *std* | AUPR | *std* |
| Enzyme | 0.88 | 0.01 | 0.11 | 0.01 | 0.92 | 0.01 | 0.29 | 0.05 |
| GPCR | 0.78 | 0.02 | 0.18 | 0.02 | 0.85 | 0.02 | 0.27 | 0.04 |
| IC | 0.85 | 0.01 | 0.25 | 0.02 | 0.89 | 0.02 | 0.43 | 0.06 |
| NR | 0.68 | 0.08 | 0.16 | 0.05 | 0.78 | 0.05 | 0.25 | 0.09 |
| Kinase | 0.85 | 0.01 | 0.32 | 0.03 | 0.86 | 0.01 | 0.36 | 0.03 |

### 9.2.6 Running times and scalability

Table 9.4 shows running times for a single test fold for the different combinations of community detection algorithm and matching method. We show results for complete test folds since optimizing the parameter values via internal CV will require five times this running time before the derived model can be applied to unseen data.

According to the results, spectral partitioning is remarkably faster than Louvain, but for *NC* matching this difference shrinks. Notably, even though *NC* matching is more expensive in itself, it compensates by the fact that this matching technique by its nature requires fewer communities, what is less expensive to discover.

In order to verify scalability of our approach we compared running times on the benchmark sets to ones with IUPHAR, which is significantly larger than any of the benchmark sets. To understand how our approach scales, we compare *relative* network sizes to relative running times. To represent network size we use both the size of the adjacency matrix (i.e. the number of vertices squared) and the number of edges. Dividing these values for IUPHAR by those for the benchmark sets indicate how much bigger, in *relative terms*, the former is than the latter (Table 9.6, left-hand side).

Table 9.6: Relation between network sizes and running times for IUPHAR and benchmark data sets for spectral partitioning and the Louvain algorithm

| Data set | Quotient IUPHAR/benchmark set | | | | | |
|---|---|---|---|---|---|---|
| | Network size | | Running times | | | |
| | | | Spectral partitioning | | Louvain algorithm | |
| | $\|V\|^2$ | $\|E\|$ | $JC_{CC}$ | $JC_{NC}$ | $JC_{CC}$ | $JC_{NC}$ |
| Enzyme | 92.03 | 82.98 | 59.27 | 34.58 | 419.11 | 219.37 |
| GPCR | 1119.3 | 894.61 | 871.62 | 528.70 | 6926.28 | 3256.17 |
| IC | 660.39 | 605.22 | 492.60 | 253.56 | 5599.97 | 2525.81 |
| NR | 17685.67 | 14467.41 | 23185.33 | 9939.64 | 131599.25 | 67565.50 |
| Kinase | 435.17 | 263.73 | 349.88 | 204.35 | 1949.62 | 1185.36 |

Relative running times, i.e. running times on IUPHAR divided by running times on the benchmark sets, is shown on the right-hand side of Table 9.6. With spectral partitioning, we see that size quotients grow faster than running times quotients, with the exception for the NR data set in the *CC* formulation, which is so small that it can be treated in less than a quarter second in this setting. Another exception is the Kinase data set, running times of which grow faster than edge count. It might have to do with the fact that Kinase is rather unbalanced, with far fewer drugs than targets. These are the cases for $JC_{CC}$ measure, for $JC_{NC}$ quotients of running times are lower than ones of network size for all sets without exceptions. However, $JC_{NC}$ setting with $m = 0$, i.e. without community discovery at all, might be not representative enough, the $JC_{CC}$ does not have

this limitation. These results imply that *the method scales, at least when using spectral partitioning*. Concerning the results with Louvain, the method does not show similar behavior. Instead, it does not scale at all using neither $JC_{CC}$ nor $JC_{NC}$. We believe this is due to the Louvain algorithm implementation that we use, since implementations of the matching procedures and the evaluation framework remain the same as for spectral partitioning.

### 9.2.7 Evaluation of the genericness

In order to evaluate the generic side of the LPBYCD approach, we also test it on data sets of a different origin than drug-target interactions: user-movie recommendations, "MovieLens", and country-language interactions, Unicode languages or "Unicodelang" (see Section 2.2 for more details). We first repeat parameter selection via internal cross-validation experiment to check if this method also works with other data, then we measure the performance ceiling by optimizing parameters on test data and compare the results with state of the art methods.

#### Performance on generic data sets

As in the setting of Section 9.2.2, we optimize the thresholding method and the value of $m$ for spectral partitioning and resolution limit for the Louvain algorithm. The grid search range is same as before: $m$ is varied in the interval $[1,25]$ and resolution in $[0.1,1]$ with steps 0.1. The only exception is that for $JC_{CC}$ on the Unicodelang network, the interval was extended by $[1,75]$, otherwise the local optimum for this setting could not be found (Table 6 in Appendix 1 shows optimal $m$ values for optimal parameter setting in each fold, for some other thresholding methods optimal values of $m$ are even higher than those shown in the table). The results are summarized in Table 9.7, 9.8 for spectral partitioning and Louvain respectively.

Table 9.7: Spectral partitioning parameter optimization via internal cross-validation on generic data sets

| Data set | Measure | Internal validation | | | | External validation | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | *std* | AUPR | *std* | AUC | *std* | AUPR | *std* |
| MovieLens | $JC_{CC}$ | 0.82 | 0.00 | 0.22 | 0.02 | 0.82 | 0.01 | 0.26 | 0.02 |
| | $JC_{NC}$ | 0.88 | 0.00 | 0.30 | 0.01 | 0.89 | 0.00 | 0.37 | 0.01 |
| Unicodelang | $JC_{CC}$ | 0.69 | 0.01 | 0.04 | 0.01 | 0.68 | 0.03 | 0.02 | 0.01 |
| | $JC_{NC}$ | 0.78 | 0.01 | 0.14 | 0.01 | 0.78 | 0.01 | 0.17 | 0.05 |

Table 9.8: Louvain algorithm resolution optimization via internal cross-validation on generic data sets

| Data set | Measure | Internal validation | | | | External validation | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | *std* | AUPR | *std* | AUC | *std* | AUPR | *std* |
| MovieLens | $JC_{CC}$ | 0.79 | 0.01 | 0.18 | 0.00 | 0.77 | 0.01 | 0.22 | 0.01 |
| | $JC_{NC}$ | 0.88 | 0.00 | 0.26 | 0.00 | 0.88 | 0.00 | 0.34 | 0.01 |
| Unicodelang | $JC_{CC}$ | 0.73 | 0.00 | 0.09 | 0.01 | 0.72 | 0.02 | 0.10 | 0.03 |
| | $JC_{NC}$ | 0.80 | 0.01 | 0.13 | 0.01 | 0.81 | 0.02 | 0.16 | 0.05 |

The results are similar to the ones on benchmark sets (0.89/0.37 as maximum AUC/AUPR among generic sets on MovieLens with spectral partitioning and $JC_{NC}$ measure compared to 0.92/0.26 the best performance on the benchmark data, on IC, for the same measure). *We can therefore conclude that the good results shown above are not due to the specific application domain of biological networks but transfers to other data.* In addition, one can notice that spectral partitioning outperformed Louvain (0.89/0.37 vs. 0.88/0.34 for AUC/AUPR on MovieLens with

$JC_{NC}$), which already happened before when evaluating IUPHAR. This supports the idea that *trying different community detection algorithms can help to improve results*. Finally, we show again that our auto-tuning method works. Our detailed results w.r.t. each fold, shown in Table 6, 7 in Appendix 1 for spectral partitioning and Louvain respectively, support this observation (and the fact that different folds have different optimal parameters tells us that this optimization is important). Although AUPR values on external validation are slightly better than ones validated internally, the results on external and internal validation still align very closely, thus there is no risk of over-fitting. *We can conclude again that approach parameters can be fixed internally*.

Another interesting observation, which cannot be seen from the table but appears in our logs. Different thresholding methods behave differently when the number of eigenvalues becomes higher: "local sum" on Unicodelang with *CC* matching is not working *at all*, "personal sum" starts working *only* at $m = 46$, and "default" requires a *higher* number of eigenvalues to reach the optimum comparing to others ($m = 62$ is optimal for "default", while "personal median" outperforms it on $m = 49$). These are all results that we have not observed before. In addition, despite the fact that MovieLens is relatively bigger than Unicodelang in terms of both the number of vertices and edges, it requires a much smaller number of eigenvalues compared to Unicodelang *for both matchings* to reach the optimum (5/11 vs. 10/55 as average *m* for MovieLens and Unicodelang with *NC*/*CC* matching respectively). These results let us conclude that *trying different settings have marked impact on results, and our auto-tuning method can help to find an optimal parameter setting by automatically optimizing parameters on the test data*.

#### State of the art comparison

In addition to evaluating the results internally, we optimize our approach on the test data to get an idea of maximum attainable performance and we compare the results with the state of the art methods from [28]: ALADIN, BLM-NII [114], WNN-GIP [160] and NetLapRLS [166].

As evaluation protocol we use 5x5 fold cross-validation as described in Section 9.1.2, which allows us to compare running times with other methods. We optimize m in the interval [1,10] for spectral partitioning, other parameters remain unchanged: 9 different thresholds for spectral partitioning and resolution limit from 0.1 to 1 with steps 0.1 for Louvain. In addition to standard quality measures – averaged AUC, AUPR and their std values – we measured both *cpu* time and *exact* time to assess computational complexity of the methods[1]. As implementation of the state of the art, the toolbox from [28] was used. The experiments are run on a server with 2 processor units of class Intel Xeon E5-2680 with 48 cores in total and 512 Gb of RAM.

The results presented in Table 9.9 demonstrate that ALADIN has best AUC and AUPR on both sets, and NetLapRLS is the second best. Spectral partitioning with $JC_{NC}$, Louvain with $JC_{NC}$, and BLM-NII go next in this comparison with very similar results. On the MovieLens data set, all of the three have same AUC, but first two are better on AUPR. On Unicodelang, the behavior of the methods is not much different from that: BLM-NII has better AUC than spectral partitioning ($JC_{NC}$) and Louvain ($JC_{NC}$), but the latter two again improve on AUPR. The fact that these three methods have approximately similar performance is not surprising. BLM-NII uses a bipartite local model, which is implemented in $JC_{NC}$ measure used by our methods to compute the average between predictions, with only difference is that predictions themselves for BLM-NII and our methods are derived in a different manner. When it comes to the cpu time comparison, the two best performing methods, ALADIN and NetLapRLS, become by an order of magnitude slower than Louvain ($JC_{NC}$). On the MovieLens data set, NetLapRLS is slower than spectral partitioning ($JC_{NC}$), but on Unicodelang becomes even slower by an order of magnitude. BLM-NII goes from second fastest on MovieLens to fourth fastest on Unicodelang. WNN-GIP has the worst performance and remains the slowest. We use cpu time to illustrate total complexity of each method implementation

[1] the difference is that cpu time gives more accurate estimation when using multi-thread computations

Table 9.9: The LPBYCD performance comparison with state of the art methods on generic data sets

| Data set | Method | Performance | | | | Time, s | |
|---|---|---|---|---|---|---|---|
| | | AUC | *std* | AUPR | *std* | cpu | exact |
| MovieLens | Spectral part. ($JC_{NC}$) | 0.89 | 0.00 | 0.38 | 0.00 | 6333787 | 839696 |
| | Louvain ($JC_{NC}$) | 0.89 | 0.00 | 0.34 | 0.01 | 116486 | 116676 |
| | ALADIN[a] | 0.93 | 0.00 | 0.51 | 0.00 | 3872400[a] | 16819260[a] |
| | BLM-NII | 0.89 | 0.01 | 0.29 | 0.02 | 2191787 | 72120 |
| | WNN-GIP[a] | 0.78 | 0.07 | 0.21 | 0.08 | 13200799[a] | 546600[a] |
| | NetLapRLS[a] | 0.92 | 0.00 | 0.47 | 0.00 | 4562345[a] | 61500[a] |
| Unicodelang | Spectral part. ($JC_{NC}$) | 0.79 | 0.02 | 0.17 | 0.02 | 329217 | 16000 |
| | Louvain ($JC_{NC}$) | 0.80 | 0.01 | 0.17 | 0.03 | 3193 | 3193 |
| | ALADIN[b] | 0.85[b] | 0.01 | 0.20[b] | 0.03 | 89820 | 1945980 |
| | BLM-NII[b] | 0.83[b] | 0.01 | 0.07[b] | 0.00 | 824250 | 20100 |
| | WNN-GIP | 0.70 | 0.01 | 0.06 | 0.01 | 1478771 | 42720 |
| | NetLapRLS[b] | 0.83[b] | 0.01 | 0.20[b] | 0.02 | 1272805 | 25920 |

[a] quick optimization[4]
[b] slightly modified data

when using multi-thread computations. The implementation of Louvain ($JC_{NC}$) does not use this advantage, this is why cpu time and exact time for this method remain the same. The only result which is surprising, the exact time of ALADIN is much higher than its cpu time. It can only gives us a conclusion that cpu time measured by the toolbox is not correct for this method.

One common limitation the ALADIN, NetLapRLS and BLM-NII methods have, their implementation requires an input with full similarity matrix for vertices. Our data often do not have such similarity information. When an item in the data do not have a value for any of the attributes we fill similarity of pairs with such item by zeros, which is considered to be the same in our interpretation[2]. As a result, ALADIN, NetLapRLS and BLM-NII fail on Unicodelang, because there is not enough positive similarity information, and to overcome it we have to perform experiments on slightly modified data[3]). Another issue we faced with ALADIN, NetLapRLS and WNN-GIP on the MovieLens data set, we managed to run these methods in quick optimization mode only[4]) due to incredibly high exact running time in the "full" mode. Taking into account that MovieLens is a bigger set than Unicode by the number of both vertices and edges (Table 2.4), we suspect that this issue might happen because given methods (or their implementations) are not scalable enough.

To sum up, we can conclude that *the* LPBYCD *approach based on the Louvain algorithm with* $JC_{NC}$ *measure provides the best trade-off in this experiment*. Being faster than the rest methods, it has no limitations regarding the input data and its predictive results are close to the maximum.

### 9.2.8 Interpretability

The LPBYCD approach offers a *straightforward option for interpretation of a link prediction*. In addition to basic information such as ranking or optimal parameter settings, we can show for each of the two vertices in the network the following:

1. the communities they belong to (with possibility to explore all nessesary information about communities: their sizes, nature (were they mixed or pure), identifiers of other drugs and

[2] by doing that we try to preserve similarities of items with available attributes rather than to help to infer similarities for the items without any attribute

[3] we replace the default similarity value 0.0 for such items without properties by a default similarity value 0.0001

[4] undocumented feature available in the toolbox of [28]

targets in the communities),
2. the weights of intra-community edges,
3. the number and layout of inter-community edges,
4. their numerical translation by the measure and matching technique (which equates to a link probability in the case of a normalized measure, with the JC for instance).

To illustrate how it works, consider the same example as in Section 8.2.4. In that example, the pair of drug $l2$ and target $t2$ is predicted to be interacted. The following information can be presented in addition to this result:

1. drug $l2$ belongs to community $C1$ (size 3, other drugs: $l1$, $l3$) and target $t2$ to community $C3$ (size 2, other targets: $t1$);
2. in this simplified example, there is no similarity networks, thus the vertices inside communities are not directly connected;
3. there are 3 links between $C1$ and $C3$ shown as in Fig. 8.9;
4. in case of $JC_{CC}$ measure the probability of interaction for the predicted pair is 0.5, and in case of $JC_{NC}$ is 0.42.

This is a possibility that is not available for recent, well-performing techniques based on latent models, such as graph embedding. This kind of information *might be useful for a domain expert* (drug researcher in case of drug-target prediction), who will be the user of the approach. It should help to get a *better understanding and/or more confidence in the prediction results*.

## 9.3 Validation of link prediction with external knowledge

In this section, I present results on the evaluation of the LPBYCD approach using the UMLS as an external resource, the setting described in Chapter 3.

### 9.3.1 Experimental settings

In this experiment, we evaluate the quality of predictions made by LPBYCD using spectral partitioning and the Louvain algorithm as community detection techniques. The evaluation is performed using the same principle as in Section 6.4, where we assessed the quality of predictions made by NEWERMINE: predictions are performed on the full 6 layer graph of IUPHAR using the full data and predicting edges that are not present in the data, and the quality of predictions is measured by the UMLS using the framework from Chapter 3.

We run spectral partitioning and the Louvain algorithm in combination with $JC_{CC}$ and $JC_{NC}$ as best performing measures and optimal parameter settings determined in Section 9.2.5 (see Table 9.4 for the summary of parameter values). In the fashion of NEWERMINE, we run the prediction process for each drug in the set and use precision at 20 to make new predictions. As a result, the final number of predictions remain the same for all methods, which allows us to make a fair comparison with NEWERMINE and between methods themselves.

The rest of the experimental settings remain the same as in Section 6.4.1: the *2018AB* release of UMLS is used as an external knowledge base, the framework described in Section 3.2.1 in combination with similarity measures from Section 3.1.3 is used for result evaluation, and precision, and the normalized version of precision are used to assess the quality of predictions.

### 9.3.2 Experimental results

The results for each method and measure are presented Table 9.10. The non-normalized values of precision correlate with the ones from Section 6.4.2: the Non-zero score provides higher values, Semantic vectors lower values, while Path and Lin similarities smooth out this difference (see Table 6.5 for comparison).

Table 9.10: Precision achieved using spectral partitioning and the Louvain algorithm with optimal parameter settings, and verified using UMLS with different similarity measures

| Approach | Path similarity | | Lin similarity | | Semantic vector | | Non-zero score | |
|---|---|---|---|---|---|---|---|---|
| | *Prec* | $Prec^{norm}$ | *Prec* | $Prec^{norm}$ | *Prec* | $Prec^{norm}$ | *Prec* | $Prec^{norm}$ |
| Spectral part. ($JC_{CC}$) | 0.16 | 0.66 | 0.00 | 0.24 | 0.00 | 0.08 | 0.61 | 0.78 |
| Spectral part. ($JC_{NC}$) | 0.14 | 0.62 | 0.01 | 0.65 | 0.00 | 0.00 | 0.36 | 0.46 |
| Louvain ($JC_{CC}$) | 0.16 | 0.69 | 0.00 | 0.20 | 0.00 | 0.52 | 0.81 | 1.00 |
| Louvain ($JC_{NC}$) | 0.14 | 0.59 | 0.00 | 0.38 | 0.00 | 0.04 | 0.42 | 0.54 |

Maximum response values remain the same as in Table 6.6 for all methods since the same drugs and targets were used from IUPHAR to compute it. The normalization process balances difference at precision for the different methods, at least for the Path similarity measure. For other measures, the result is rather unexpected, with Semantic vectors in particular having differences in values more than 50% between different prediction methods. In addition, the Non-zero score reaches its limit in the normalized version of precision, which also never happened before. It even makes it challenging to determine the best performing method: according to Path, Semantic vectors and Non-zero this should be Louvain with the $JC_{CC}$ measure, but according to Lin spectral partitioning with $JC_{NC}$. As for the second best, the result of Lin also differs from the others: Path, Semantic vectors and the Non-zero score point to spectral partitioning with $JC_{CC}$, but Lin to Louvain with $JC_{NC}$. Finally, all measures select different methods as the third best in this comparison: Path - spectral partitioning with $JC_{NC}$, Lin – spectral partitioning with $JC_{CC}$, Semantic vectors and the Non-zero score – Louvain with $JC_{NC}$. To smooth out this difference and determine which approach most like to be the best, we take an average of normalized precision values. Similar to Table 6.5, we use the median function to remove outliers.

Table 9.11: Percentage of predictions made by our method using spectral partitioning and the Louvain algorithm with optimal parameter settings, verified by UMLS, and averaged among different similarity measures

| Approach | Avg (median) $Prec^{norm}$ % |
|---|---|
| Spectral part. ($JC_{CC}$) | 44.96 |
| Spectral part. ($JC_{NC}$) | 53.82 |
| Louvain ($JC_{CC}$) | 60.61 |
| Louvain ($JC_{NC}$) | 45.89 |

According to the results shown in Table 9.11, Louvain with $JC_{CC}$ is the best performing combination, and spectral partitioning with $JC_{NC}$ the second best. The combination of Louvain with $JC_{NC}$, which was a favorite in other experiments with other data sets comes in only the third. Concerning the comparison with NEWERMINE, surprisingly enough, the latter outperforms any of community detection based methods with 83% of verified interactions (see Section 6.4.2).

To sum up, the best result that can be achieved using our approach is one of the Louvain algorithm in combination with $JC_{CC}$ measure. According to this, almost 61% of interactions can be verified on the UMLS taking into account its limitations. From which we can draw a conclusion that *the method has an acceptable performance, and one can use our approach to make predictions on new unseen data*.

## 9.4 Conclusion

We have presented a framework for link prediction in bipartite multi-layer graphs using graph community structure and link prediction measures adapted from those proposed in the literature. We have found empirically that combining the well-known and relatively straightforward Jaccard coefficient, particularly in a BLM formulation, with the Louvain algorithm for community detection allows us to achieve results that are competitive with the state of the art. We have also demonstrated that it is possible to set the parameter values of the community detection techniques via internal cross-validation and that they transfer well to unseen data.

We also evaluated our approach on the much larger and sparser IUPHAR data set. Using those data, we have verified predicted interactions in terms of their biological semantics, shown that external validation aligns well with validation using known labels in test data, and assessed scalability and interpretability.

Finally, we tested our approach on data sets of which origin is different from drug-target interaction. We demonstrated experimentally that our approach performs well regardless of the domain of the task. Comparison with state of the art showed that our approach is more general and does not depend on the structure of data, and can be the fastest comparing to other methods, depending on the employed community detection approach.

# IV Detecting promiscuous compounds


**10 Identification and characterization of promiscuous compounds** 117
10.1 Introduction
10.2 Related work
10.3 Substructure mining and model learning
10.4 Experimental evaluation
10.5 Graphical interface
10.6 Conclusion

# 10. Identification and characterization of promiscuous compounds

*In this chapter, I deal with the problem of detection and elimination of Pan Assay Interference Compounds (PAINS), a significant problem in the drug development process described in the introduction of this thesis. I start with the introduction to the problem, which includes motivation, context and definitions, and the description of the idea of an approach. I then present the state of the art on the problem and introduce some techniques I base my approach on. Next, I present the approach that includes graph mining, sampling and predictive model construction parts. This is followed by the experimental evaluation including data description and data preparation, description of the experimental setup and the results with different settings. Finally, I present the graphical interface that I developed for experts in chemistry for making predictions and exploring the results. At the end, I finish the chapter with the conclusion. The approach and the graphical interface called* PREPEP *are the main contribution in this chapter, also published at KDD'18, ECML-PKDD'20 and presented at SFCi'17 (the French National Society in chemoinformatics).*

## 10.1 Introduction

In this section, I explain the importance of the problem of detecting and eliminating of PAINS from the drug development process and provide motivation to design a new approach for doing that. I then describe the context in more detail and formally defined some important concepts used in this chapter. At the end, I briefly describe our idea on how combining data mining and machine learning can help to solve the given problem.

### 10.1.1 Motivation

The motivation for this work origins from the drug development process described in the introduction of this thesis. In one of the first steps of this process, high-throughput screening (HTS) is performed to find compounds showing activity regarding a therapeutic target or its surrogates (so-called "hits"). In this work, we deal with a problem which appears at this step. Certain molecules can emerge as hits that do not actually interact with the target. Such false positives are impossible to optimize to get a therapeutic effect and therefore do not lead to successful drug development. In other words, the whole following process might be ruined, wasting time and resources. To avoid these situations, such molecules must be detected and excluded from drug development process at an early stage.

This problem has been identified in the literature [9], and subsequent studies have shown their repeated presence in publications introducing supposed leads [122, 133]. A recent joint editorial of the editors-in-chief of journals of the American Chemical Society [3] has drawn special attention to this problem. A number of structural alerts were introduced in [9], based on analysis of a proprietary compound database, which have found widespread use since then.

Recent work has shown that existing structural alerts are not up to the task of identifying those promiscous compounds, which had already been named PAINS, and that an alternative method for their identification is therefore needed [32, 84]. To address this short-coming, we combined machine learning and data mining techniques and proposed a new method for PAINS identification based on that. We validated our goals and designed choices with our partners and co-authors from ICOA (l'Institut de Chimie Organique et Analytique) and CERMN, two medicinal chemistry laboratories in Orléans and Caen, respectively, and evaluated the approach on recent benchmark data sets from the literature. In addition, we developed a graphical interface allowing not only to perform predictions, but also visually explore the reasons which lead to certain results, a functionality, which might be useful for drug developer experts.

### 10.1.2 Context/Definitions

As described in the introduction of this thesis, modern drug development can be summarized as a four-step process:

1. high-throughput screening (HTS) to find compounds showing activity regarding a therapeutic target or its surrogates (so-called "hits"),
2. more fine-grained evaluation and limited optimization to identify promising "leads",
3. additional optimization and pre-clinical testing,
4. clinical testing.

Depending on the test readout used during the first step, certain molecules can show non-target specific activity. A particularity of non-specific interaction is that such compounds seem to show activity in different, potentially independent biochemical assays, hence their name "frequent hitters" (FHs). Obviously, FHs should be filtered out at an early step to avoid leading drug developers wasting their time and resources.

In the following, we formally define the concepts that we have just informally introduced. This includes definition of such concepts as *hits*, *frequent hitters* (FHs), *infrequent hitters* (iFHs), but first we introduce the notations which will be used in our definitions.

**Definition 10.1.1 — Molecule representation.** Let $\mathscr{M}$ be a set of molecules, with each $m \in \mathscr{M}$ of the form $(x, \mathbf{y})$, x being the representation of the molecule and $\mathbf{y} \in \{0,1\}^d$ its activity profile.

Note, that we do not concretely specify the molecule representation – different options will be discussed in Section 10.4.1. The activity profile results from the (post-processed) outcomes of different HTS assays, with $d$ the number of bioassays in which a compound has been tested.

**Definition 10.1.2 — Hit.** A hit is a molecule $(x, \mathbf{y}) \in \mathscr{M} : ||\mathbf{y}|| > 0$.

**Definition 10.1.3 — Frequent hitter.** A frequent hitter (FH) is a molecule $(x, \mathbf{y}) \in \mathscr{M} : ||\mathbf{y}|| \geq \theta$, with $\theta$ a user-specified threshold.

**Definition 10.1.4 — Infrequent hitter.** An infrequent hitter (iFH) is a molecule $(x, \mathbf{y}) \in \mathscr{M} : ||\mathbf{y}|| < \theta$, with $\theta$ a user-specified threshold.

Those compounds with non-target specific activity, identified through FHs, have been dubbed in the literature as "**P**an-**A**ssay **IN**terference compound**S**" or just "*PAINS*" (Fig. 10.1).

The user-defined threshold $\theta$ specifies how often a compound needs to show activity to be

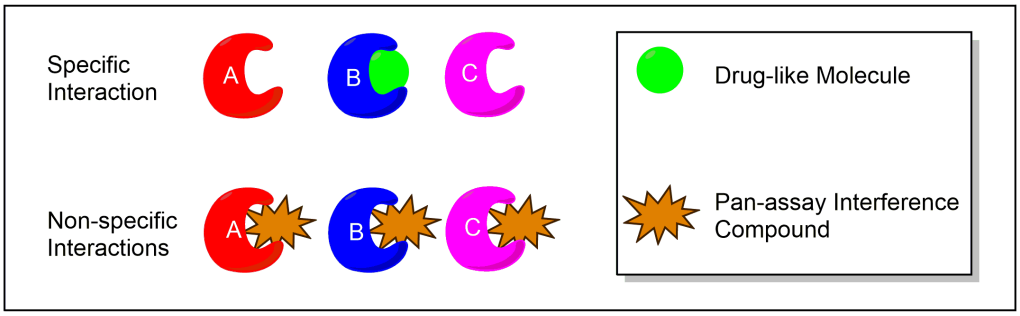


Figure 10.1: Schematic depiction of a drug-like molecule and a PAINS-like molecule. The green circle specifically interacts with target B, while the orange polygon interacts with all targets [13]

considered FH. In case of $\theta = 1$, all hits will be automatically considered FHs or *one-hitters* (1Hs), the case also studied in this chapter.

**Definition 10.1.5 — One-hitter.** One-hitter (1H) is a molecule $(x, \mathbf{y}) \in \mathscr{M} : ||\mathbf{y}|| = 1$.

When appropriate, we will refer to the subsets made up of FHs/iFHs/1Hs as $\mathscr{M}_{FH}$, $\mathscr{M}_{iFH}$, $\mathscr{M}_{1H}$, respectively.

Later on, we will speak of *Dark Chemical Matter* (DCM) i.e. compounds that have been assayed often and not showing any activity. Formally they can be defined as follows.

**Definition 10.1.6 — Dark Chemical Matter.** Dark Chemical Matter (DCM) are mole-cules $(x, \mathbf{y}) \in \mathscr{M} : ||\mathbf{y}|| = 0, d >> 10$.

In this chapter, we consider molecules as graphs, i.e. representing them in terms of their 2D structure, as described in Section 1.1.2. In this form, any given molecule can be represented as a labeled graph of which the vertices are labeled with atom names, and edges with atomic bonds. To build descriptors for molecules, we encode them by subgraphs (Def. 1.1.20) and we require of our subgraphs to be connected (Def. 1.1.22).

In the next section, we describe our idea on how we propose to perform classification of compounds into FHs and iFHs using data mining and machine learning techniques.

### 10.1.3 The idea

In this chapter, we propose to use pattern mining techniques to derive new structural alerts, which we combine with machine learning models to predict PAINS. Pattern mining techniques, taking inspiration from statistics, have always allowed to constrain the minimum empirical support or effect size as shown in [173]. Using quality measures that contrast two classes against each other, e.g. growth rate [110, 117] or statistical measures [27, 180], allows to mine substructures that are representative of FHs or iFHs.

To develop the approach, we have to address three issues:

1. the extreme imbalance of the underlying data, which by nature contains very few FH examples,
2. how to derive a small set of meaningful structural descriptors, and
3. how to create an accurate predictive model that is at the same time interpretable, to facilitate acceptance by researchers in medicinal chemistry and aid further scientific research in the field.

To solve the first, we propose a sampling technique in combination with majority vote prediction, creating balanced sets necessary for learning and covering all instances of the over-presented class. To address the second issue, we represent chemical compounds as graphs and make a use of graph

mining techniques, discriminative subgraph mining in particular. For the last, finally, we employ decision tree classifiers and design a graphical interface allowing to explore the predictions for unseen molecules as well as which subgraphs were involved in the prediction. In Section 10.3, we will describe all of the proposed techniques in more detail.

## 10.2 Related work

In this section, I briefly describe the state of the art on PAINS identification and characterization and some other techniques, Quantitative structure activity relationship (QSAR) models and pattern mining in particular, both of which are used later in Section 10.3 to describe our approach for solving given problem.

### 10.2.1 PAINS identification/characterization

The term PAINS has been introduced by Baell *et al.* in [9], the authors of which proposed a set of structural alerts to identify and exclude PAINS. Following work has mainly focused on identifying PAINS mistakenly published in the literature and identifying their mechanisms [8, 10, 44, 122, 133, 156, 157].

Criticism of Baell *et al.*'s filters is more muted, with one notable publication [116] arriving at conclusions contrary to those of [157]. Recently, two publications [32, 84] have called the usefulness of existing structural alerts into question. Capuzzi *et al.* show existing alerts *miss* a large proportion of FHs in an independently curated molecular data set, yet *match* a large proportion of iFHs, successfully marketed drugs (arguably a subset of 1Hs), and even DCM compounds. In trying to understand this behaviour, they illustrate that many (190 of 480) of the original alerts have been derived from a single compound each.

Alternatives are, to the best of our knowledge, rare so far, with the exception of [174], which proposes tagging compounds with different *scaffolds*, each of which has a Bayesian-inspired FH score. There also exist some other tools for FH prediction [113, 151], both in the form of web services. The first is based on extremely randomized trees and the latter uses deep learning, each of which significantly limits result interpretability.

### 10.2.2 (Q)SAR modeling

(Quantitative) structure activity relationship modeling is concerned with using statistical or machine learning models to predict a compounds' activity w.r.t. a given target, based on its structural properties. Work in this field is well-established [153] and on-going [43, 164].

Several authors working in this direction [68, 86] have drawn attention to the importance of the *interpretability* of derived models – black box classifiers do not help in isolating and optimizing effective components – and to the unavoidable trade-off with accuracy that will result from it. (Q)SAR methods typically assume a previously defined therapeutic target for which bioassay data exists.

### 10.2.3 Pattern mining

Transactional graph mining has often been motivated with (frequent) substructure discovery in molecular data. The probably best-known algorithm is gSpan [173] but there exist alternatives [81, 95, 126], and experimental studies [125, 165] have shown inconclusive results regarding their relative merits. By using the result presented in [119], which exploits the convexity of certain quality measures, such approaches can be used to find discriminative substructures for the (Q)SAR problem [27, 180]. Alternatively, one can use the classic *frequent* pattern mining approach and combine the results to arrive at discriminative *conjunctions* of patterns [110, 117].

## 10.3 Substructure mining and model learning

As mentioned in Section 10.1.3, the available data is highly imbalanced, a problem that needs to be addressed to enable feature construction and model learning. In addition, as we will show later in Section 10.4.1, there are plentiful options for molecule representation: by graphs, chemical fingerprints, numerical descriptors and their combinations. Taking into account only the number of fingerprints – $107k$ – a curse-of-dimensionality issue might be faced, as well as any learning approach used for inducing a model might be significantly slowed down. Finally, the model itself should be interpretable, both to convince experts to *accept* the predictions and to facilitate further scientific progress on the application side. In this section, I discuss how I address those three challenges while developing a new approach.

### 10.3.1 Discriminative subgraph mining

Chemical fingerprints are derived from chemical background knowledge and usually based on a very low minimum support in the data, hence their large number. Quality measures that score how well subgraphs discriminate between two classes, on the other hand, can capture much more information in a smaller set of patterns. We therefore chose to use GSPAN, augmented with the capability to use the *information gain* measure via upper-bound pruning, to mine discriminative subgraphs. We define information gain through entropy.

**Definition 10.3.1 — Entropy.** Let the data set $\mathscr{M}$ consist of two subsets $\mathscr{M}_{FH}, \mathscr{M}_{iFH} : \mathscr{M}_{FH} \cap \mathscr{M}_{iFH} = \emptyset \wedge \mathscr{M}_{FH} \cup \mathscr{M}_{iFH} = \mathscr{M}$. The entropy of the data set is:

$$H(\mathscr{M}) = -\frac{|\mathscr{M}_{FH}|}{|\mathscr{M}|} \cdot \log_2 \frac{|\mathscr{M}_{FH}|}{|\mathscr{M}|} - \frac{|\mathscr{M}_{iFH}|}{|\mathscr{M}|} \cdot \log_2 \frac{|\mathscr{M}_{iFH}|}{|\mathscr{M}|}.$$

Now using the definition of entropy we can define information gain.

**Definition 10.3.2 — Information gain.** Given $\mathscr{M}, \mathscr{M}_{FH}, \mathscr{M}_{iFH} : \mathscr{M}_{FH} \cap \mathscr{M}_{iFH} = \emptyset \wedge \mathscr{M}_{FH} \cup \mathscr{M}_{iFH} = \mathscr{M}$, the information gain of a subgraph $G$ w.r.t. $\mathscr{M}$:

$$IG(G,\mathscr{M}) = H(\mathscr{M}) - \frac{supp(G,\mathscr{M})}{|\mathscr{M}|} \cdot H(cov(G,\mathscr{M})) - \frac{|\mathscr{M}| - supp(G,\mathscr{M})}{|\mathscr{M}|} \cdot H(\mathscr{M} \setminus cov(G,\mathscr{M})).$$

Information gain rewards subgraphs that change the distribution in the covered and uncovered subsets towards one of the two classes. Our hypothesis is that there are shared structural characteristics of FHs that can be quantified via information gain, whereas iFHs are too heterogeneous for shared patterns to appear. There are other convex quality measures that could be chosen but there are no well-founded selection criteria, apart from the fact that normalized measures, like information gain, are preferable to unnormalized ones, such as $\chi^2$, which overly reward patterns with very small effects if the effect size is large.

The decision for GSPAN is mainly a practical one: with an implementation for discriminative pattern mining provided by Siegfried Nijssen, there was no need to implement a miner ourselves. The decision to mine *graphs* instead of *sequences*, which work as well and are easier to mine [27], was deliberate, however: when presenting results to domain experts, graph patterns, particularly those including cycles, are more informative. We represented all molecules in the form of hydrogen-suppressed graphs, i.e. all vertices labeled with "H" were removed from the graphs. We have mined the top-100 subgraphs according to information gain.

### 10.3.2 Balancing the data

The data set, which will be described in more detail in Section 10.4.1, has size $|\mathscr{M}| = 153539$, of which only a very small subset are FHs: $|\mathscr{M}_{FH}| = 902$ when using the threshold $\theta = 2$ (meaning

that molecules are active on at least two assays). Such an imbalance will have a negative impact both on subgraph mining, and on model learning. On subgraph mining because subgraphs that split off a certain amount of iFHs while present in a *majority* of iFHs and all FHs would receive elevated information gain scores without being representative of FHs. On model learning because a model that always predicts the majority class, i.e. iFHs, will achieve an accuracy of 99.41%.

The recommended solution for such a setting is to undersample the majority class to create a balanced data set. In our case, creating a *single* undersample would remove the overwhelming majority of iFHs, without any guidance regarding which instances to keep. We elected therefore to perform repeated sampling, using *all* FHs every time and randomly sampling an equal amount of iFHs from the data. To ensure that we use every iFH at least once, we need at least 171 samples but since we would like to contrast different combinations of iFHs against the FHs in the data, we scaled up to 200 samples.

### 10.3.3 The predictive model

The task that the predictive model is supposed to address is a form of SAR – instead of having a single therapeutic target, the target class is that of frequent hitters, i.e. molecules that show activity regarding *different* targets. The general setting remains the same, however: *given* molecules described in terms of their structure and labeled by an activity indicator, *find* a model that reliably predicts the latter based on the former.

As indicated in Section 10.2, an important aspect of SAR models is interpretability. Experts are more likely to accept predictions when they are presented with explanations for those predictions in a language they understand, an issue not limited to the SAR setting [142], and SAR models are, after all, only a means to an end, that end being an understanding of biochemical mechanisms and further scientific progress in the domain.

The interpretability constraint rules out powerful non-symbolic learners such as support vector machines [42], or neural networks, particularly those used in deep learning approaches [144]. Decision tree learners are prime representatives of interpretable models, even if using them would theoretically trade-off accuracy against interpretability. Practically, on the other hand, combining decision trees into an ensemble using *Bagging* [25] has been shown to lead to very good results. The resampling described in the preceding section is similar to Bagging – not identical, because the instances of one class remain fixed – so we would expect good performance of an ensemble of decision tree classifiers as well.

The full workflow is therefore as follows:

1. we create 200 data sets with equal amounts of FHs and iFHs by sampling from $\mathscr{M}_{FH}$,
2. we mine the top-100 discriminative subgraphs from each data set,
3. we encode each molecule with the numerical descriptors described in Section 10.4.1 and a hundred-dimensional bit-vector indicating the presence/absence of the subgraphs mined from this data set,
4. we learn one decision tree classifier (DT) for each data set.

An unseen molecule is encoded in terms of its numerical descriptors and the discriminative subgraphs for each of the 200 training sets and classified by the respective decision trees. The final classification is derived as a majority vote of all 200 DTs' predictions, in case of a tie the majority class (iFH) is predicted. A schema of the process is shown in Fig. 10.2.

## 10.4 Experimental evaluation

In this section, I evaluate the quality of the model described in the previous section. Concretely, the first question is which representation reveals itself to be most beneficial for the prediction task, and whether decision tree classifiers can, as we have reasoned above, deliver acceptable quality for

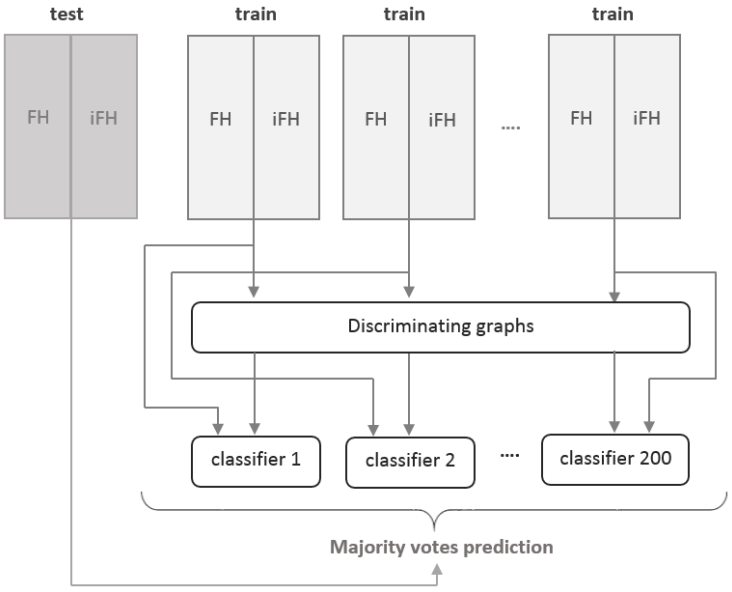


Figure 10.2: Schema of the model construction workflow

FH prediction in a well-curated data set. I then evaluate whether the model's performance transfers to other compounds, and compare to the PAINS alerts criticized by Capuzzi *et al.*.

### 10.4.1 Data preparation

The data we base our model on have been curated by Capuzzi *et al.* for their publication [32]. As usual in that community, the data are publicly available in the form of PubChem IDs and SMILES strings [80] at `http://pubs.acs.org/doi/abs/10.1021/acs.jcim.6b00465`. Following Baell *et al.* [9], each compound in the data set has been evaluated in six assays, and the activity threshold for declaring a compound FH has been set to $\theta = 2$.

Molecules were standardized using the tool VSPrpep [59], giving rise to a graph representation. Starting from that representation, each chemical compound was described using the RDKit Morgan fingerprint implemented in Knime [16] with a radius of three chemical bonds [138] as used in a previous study [23]. The fingerprint composed of 0 and 1 bits takes into account the neighborhood of each atom by exploring the adjacent atoms in a three connected bounds. This resulted in the representation of each molecule by a bitstring with the size of 107k bits each. Numerical molecular 0D-descriptors were derived using Pipeline Pilot [17], giving rise to 192 numerical descriptors. In other words, each compound is available in the form of a graph, a bitstring of fingerprints, and a set of numerical descriptors.

Additional data that we use for evaluation is available in graph form[1].

### 10.4.2 Experimental setup

In total, we have used three data sets, all proposed by Capuzzi *et al.* in [32]:

1. The data set described in the previous section, comprised of 902 FHs and 152,637 iFHs. We use these data for the proof-of-concept experiment that evaluates whether a model accurately predicting FHs and iFHs can be learned.
2. A data set of 73,333 compounds that have each been evaluated in at least 25 separate bioassays, randomly sampled from the PubChem database [90]. The data set is separated into those compounds containing PAINS alerts as originally defined in [9], hereafter referred

[1] All data can be downloaded at: `https://zimmermanna.users.greyc.fr/supplementary-material.html`

Table 10.1: Evaluation of the performance of the majority vote decision tree model for different feature sets and different ratios of FHs and iFHs in the test data

| Setting | Features | Accuracy | | AUC score | | Precision | | Recall | |
|---|---|---|---|---|---|---|---|---|---|
| | | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ |
| Balanced | subgraphs | 80.00 | 16.67 | 0.80 | 0.17 | 0.94 | 0.19 | 0.70 | 0.04 |
| | numeric | 84.89 | 3.06 | 0.85 | 0.03 | 0.86 | 0.04 | 0.84 | 0.04 |
| | subgraphs+numeric | 85.39 | 14.96 | 0.85 | 0.15 | 0.89 | 0.15 | 0.87 | 0.04 |
| Slight imbalance | subgraphs | 88.17 | 28.68 | 0.79 | 0.16 | 0.91 | 0.29 | 0.69 | 0.05 |
| | numeric | 84.18 | 0.95 | 0.84 | 0.02 | 0.35 | 0.02 | 0.85 | 0.03 |
| | subgraphs+numeric | 83.96 | 26.80 | 0.85 | 0.15 | 0.51 | 0.16 | 0.85 | 0.03 |
| Severe imbalance | subgraphs | 89.81 | 31.28 | 0.80 | 0.15 | 0.90 | 0.31 | 0.71 | 0.05 |
| | numeric | 84.36 | 0.65 | 0.84 | 0.03 | 0.05 | 0.003 | 0.84 | 0.06 |
| | subgraphs+numeric | 84.59 | 29.43 | 0.85 | 0.14 | 0.12 | 0.05 | 0.86 | 0.05 |

to as *Random-PAINS* (14,611 compounds), and those not containing PAINS alerts (*Random-NoPAINS*, 58,722 compounds). These data serve two purposes: on the one hand they allow us to evaluate whether our model presents a clear alternative to the PAINS alerts proposed in [9]. On the other hand, we will assess whether a model derived from a particular bioassay transfers to other settings.

3. A data set of 3570 DCM compounds (tested in at least 100 bioassays) that contain PAINS alerts. Those data should ideally *never* be classified as active, let alone FH, something that existing PAINS alerts fail to do, and that we expect our model to be able to do.

As mentioned above, we used Siegfried Nijssen's GSPAN implementation for discriminative subgraph mining for the graph mining step. Data sampling was implemented in Python 2.7, using the networkx library [71] (version 1.11) for subgraph matching, and scikit-learn's [131] (version 0.17.1) decision tree learner, an implementation of CART [26]. After internal validation, minimum leaf size for decision trees was fixed to 3% of training data.

### 10.4.3 Performance evaluation for FH classification

To assess the model's performance, we performed three different ten-fold cross-validations on the data set comprised of FHs and iFHs: $\mathscr{M}_{FH}$ was separated into 10 folds, and for each fold a corresponding number of iFHs sampled. We evaluated three settings for the test fold:

1. **balanced data**, 90/91 FHs : 90/91 iFHs,
2. **slight imbalance**, 90/91 FHs : 900/910 iFHs,
3. **severe imbalance**, 90/91 FHs : 9000/9100 iFHs.

The first setting is the ideal case in which the distribution of classes in the test data is the same as in training data, whereas the third setting is much closer to a real-life setting in which iFHs will significantly outnumber FHs. For training folds, we sample 200 times 812/811 iFHs from those not contained in the test fold. As a side-effect, this means that different training samples for the imbalanced test settings will be much more redundant, and that the "independently and identically distributed" (i.i.d.) assumption underlying inductive learning is violated for the test data.

We report several common performance measures: accuracy, precision, recall and AUC (Section 1.1.3), understanding compounds classified as FH to be positive class, and ones classified as iFHs to be negative. A model that classifies very few instances as FH, e.g. by classifying almost all instances as iFHs, can achieve very good accuracy and precision, especially on imbalanced data, while failing at its final task: filtering out frequent hitters. Recall gives additional insight into the accuracy score by measuring whether a model is *general* enough to classify a large proportion of FH as FH.

#### Classifying $\mathscr{M}_{FH}$ vs $\mathscr{M}_{iFH}$

Table 10.1 shows the results for three different representations: only discriminative subgraphs, only numerical descriptors, combination of both. For each performance measure, we report the average value over all ten folds and the standard deviation. In the case of the balanced data, all three feature sets show good performance, with numerical descriptors improving the performance of subgraphs. Precision and recall are high, indicating that FHs are correctly classified and recovered. This changes as soon as we pass to the imbalanced data sets: while using only subgraphs gives similar results as before, including numerical descriptors deteriorates performance. Not only are accuracies lower than for subgraphs but precision drops precipitously while recall stays high, which means that while those models *do* classify many FHs as such, they also sweep up many iFHs and classify them incorrectly. Using only subgraphs does better at classifying iFHs as iFH but loses some coverage on the FHs in the process. Results for the latter also become more unstable while average accuracy increases compared to the balanced setting, indicating that for certain test folds the more redundant training data is not representative enough. Yet a severe deterioration as a result of the violation of the i.i.d. assumption cannot be observed.

#### Test of different classifiers

We did not only test decision trees, though. Table 8 in Appendix 2 shows evaluation results for a number of different classifiers trained in the same manner as the decision trees, and used in a majority vote. As can be seen from comparison with Table 10.1, other techniques seem to better exploit numerical features. Looking more carefully at precision and recall results, however, shows that other classifiers do *not* manage to improve on the low precision or recall values decision trees suffer for imbalanced data. Furthermore, decision trees perform consistently better when using only subgraphs and *are*, after all, more easily interpretable.

#### Classifying $\mathscr{M}_{FH}$ vs $\mathscr{M}_{1H}$

As an additional evaluation, we removed all compounds not showing activity at all and only mined on the subset containing FHs and 1Hs. This data set is significantly smaller, $|\mathscr{M}_{1H}| = 2358$, $|\mathscr{M}_{FH}| = 902$, and much less imbalanced than the full data (27.67% : 72.33%). We therefore do not subsample but mine subgraphs and learn decision trees directly, in the context of a stratified ten-fold cross-validation, i.e. enforcing the same class distribution in test as in training data.

Table 10.2: FHs vs 1Hs experiment results

| Features | Accuracy | | AUC score | | Precision | | Recall | |
|---|---|---|---|---|---|---|---|---|
| | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ |
| subgraphs | 76.52 | 23.16 | 0.66 | 0.17 | 0.91 | 0.28 | 0.41 | 0.06 |
| numeric | 69.82 | 3.18 | 0.62 | 0.03 | 0.46 | 0.06 | 0.45 | 0.05 |
| subgraphs+numeric | 72.52 | 8.80 | 0.66 | 0.07 | 0.53 | 0.11 | 0.53 | 0.06 |

As Table 10.2 shows, the trends of the imbalanced test data can also be found here: using only subgraphs gives better accuracy and much better precision. At the same time, recall is clearly lower than in the preceding experiments but this is an acceptable outcome: while we want to filter out FHs, we want to *avoid* doing this at the expense of 1Hs, which might be viable drug leads. The final realization is that contrasting FHs and 1Hs apparently gives rise to less discriminative subgraphs. This is an expected result since we would assume that the differences between compounds showing different levels of activity are less pronounced than between those showing activity and no activity at all. Based on those results, we use only discriminative subgraphs as compound representation going forward, and the entire data set of FHs and iFHs as mining/training base.

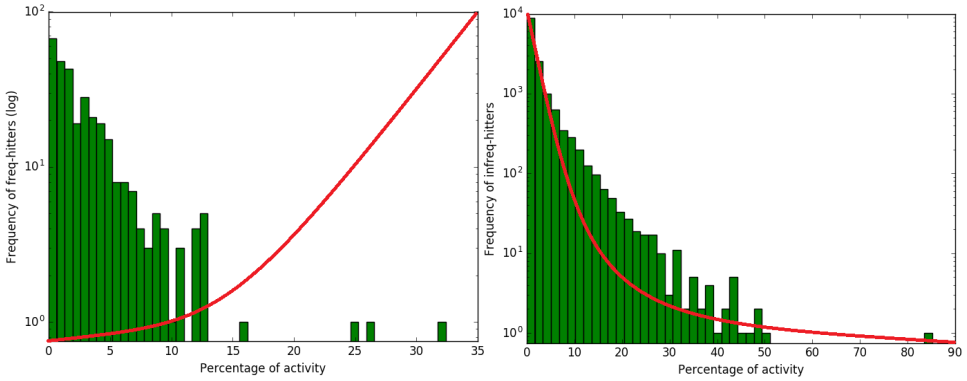


Figure 10.3: Activity histograms of Random-PAINS compounds from PubChem predicted as FH (left) and iFH (right)

### 10.4.4 Model evaluation on randomly sampled compounds

To evaluate the performance of our model on the Random-PAINS/Random-NoPAINS data set, we used the full FH/iFH data set for subgraph mining and model building. The resulting subgraphs were used to encode the test data and majority vote classification performed, as described in Section 10.3.

Table 10.3: Model evaluation on Random-PAINS, Random-NoPAINS compounds from PubChem and DCM possessing PAINS alerts

| Data | Prediction | |
|---|---|---|
| Random-PAINS | FH | 2638 |
| | iFH | 11973 |
| Random-NoPAINS | FH | 6405 |
| | iFH | 52317 |
| DCM | FH | 232 |
| | iFH | 3338 |

Rows two through five of Table 10.3 show the predictions of our model on the Random-PAINS and Random-NoPAINS subsets, respectively. The results show that the PAINS alerts proposed by Baell *et al.* and our model are very much not in agreement, i.e. we offer a clear alternative to existing filters. An additional question is, however, if our model does indeed separate frequent hitters from infrequent hitters. To this end, for the Random-PAINS classified as FH, we plot a histogram for different percentages of activity over all assays they have been tested in (Fig. 10.3, left-hand side). The same plot can be seen for Random-PAINS classified as iFH on the right-hand side of Fig. 10.3, and for Random-NoPAINS in Fig. 10.4.

Ideally, we would like compounds classified as iFH to be clustered on the left of the plots, showing low activity, and FHs to be clustered on the right. To illustrate this, we have also plotted lines in both figures, showing this ideal behavior. As we can see, our model does not yet achieve this ideal behavior. There are several possible explanations: first, the data set used for subgraph mining learning comes from a particular type of bioassay, AlphaScreen [52]. As explained in the introduction, different assays use different readout methods and offer therefore different conditions for PAINS to occur. Second, we have used a hard threshold to define FHs but a relative threshold, i.e. percentage of activity, might be more appropriate. Third, some compounds do act on several different targets and are of great interest in *polypharmacology*. Those three issues are inherent in the application setting we address here: since the mechanisms for the occurrence of PAINS are not

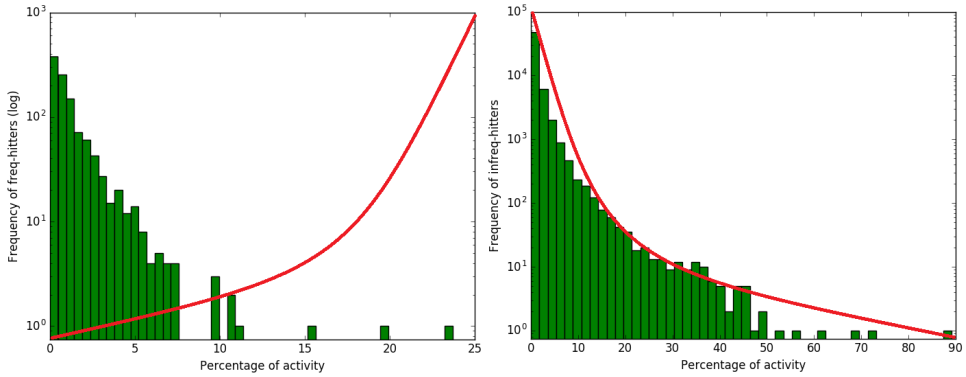


Figure 10.4: Activity histogram of Random-NoPAINS compounds from PubChem predicted as FH (left) and iFH (right)

yet clear, we cannot neatly resolve or rule out any of those effects. We intend to continue working in this direction, however.

### 10.4.5 Model evaluation on DCM

Rows six and seven of Table 10.3, finally, show the classification results of our model on Dark Chemical Matter compounds. We recall that those are compounds that over a large number of bioassays have never shown activity and should therefore not be classified as active, let alone FH. Yet all contain structural PAINS alerts as derived by Baell *et al.*. We view the fact that our model only classifies 6.5% of those compounds as FH as further evidence for its effectiveness. As in the preceding section, reducing this classification error to zero will probably require the use of data from different bioassays.

## 10.5 Graphical interface

In its current form, the PREPEP prototype[2] contains the discriminative subgraphs mined, the training data encoded in terms of those subgraphs, the code for learning the decision tree classifiers and performing the majority voting, as well as a visualization interface for test instances. When launching the application, decision trees are learned and are ready for prediction. To predict unseen data, they need to be available in the *SDF* format [18], a format widely used in the computational biology and chemistry communities, standardised in the same way as the training data. Once loaded, the unseen data is encoded in terms of the subgraphs and directly classified via majority vote. PREPEP's workflow is shown in Fig. 10.5.

The visualization uses the Python interface of the OpenBabel library [127] – the same library is also used to load SDF files. As in the case of the use of SDF files, this is due to the familiarity of the computational biology/chemistry communities with this type of visualization. The initial visualization we proposed, based on the networkx library, which inscribed discriminative subgraphs directly into the molecule graphs, was rejected by our collaborators, experts in chemistry, who are more comfortable with a visualization adhering to chemical visual standards.

A screen shot of the interface can be seen in Fig. 10.6: molecules to be predicted are shown in the left column. For each molecule predicted as FH, such as molecule 3 in this case, the right-hand column lists the discriminative subgraphs present in this molecule. Subgraphs are sorted by how often they were part of the result set of the mining operation, in descending order. Selecting a

[2] Available at: `https://zimmermanna.users.greyc.fr/software/prepep.zip`

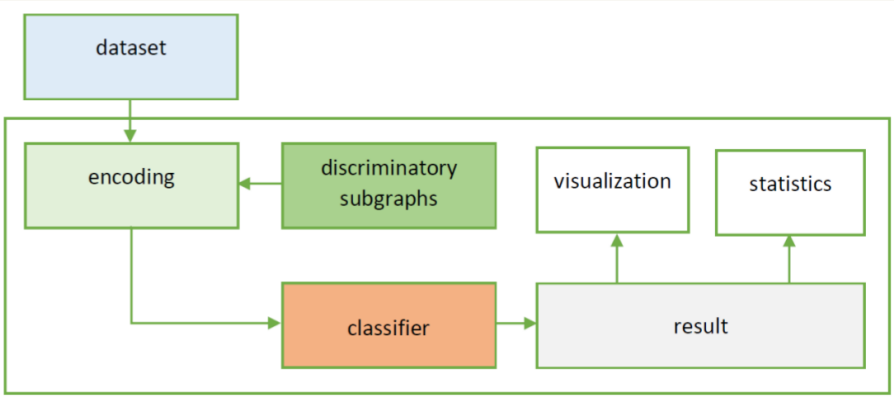


Figure 10.5: Schema of PREPEP's workflow

molecule visualizes it using OpenBabel, selecting a subgraph indicates its frequency in result sets at the bottom of the right-hand column, and visualizes the subgraph.

The work on PrePeP is ongoing. We intend to finalize the prototype and to make the tool available to the wider community.

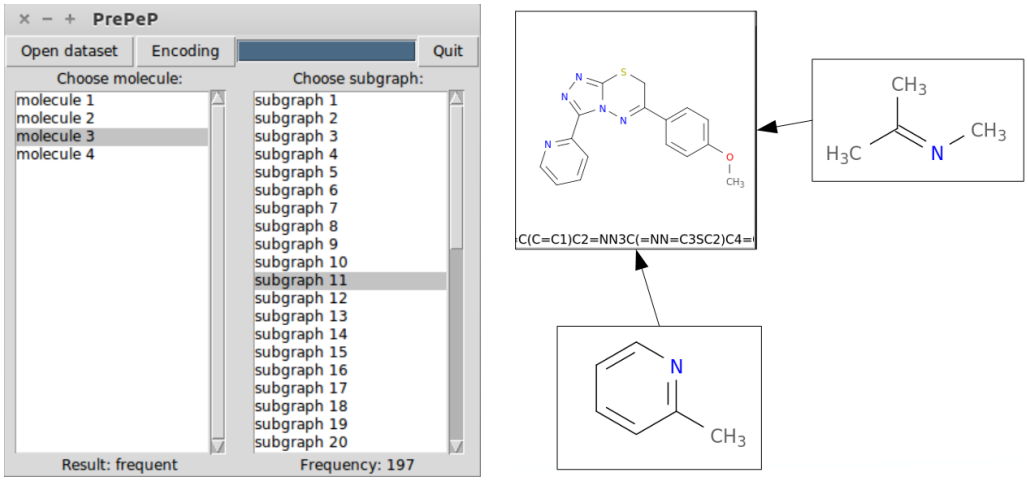


Figure 10.6: Screen shot of PREPEP's interface

## 10.6 Conclusion

We developed an approach for detection and elimination of Pan Assay Interference Compounds. The experimental results validate our approach and our design choices. As we have shown on the FH/iFH data set, it is indeed possible to build a model that predicts FHs representing PAINS with good accuracy, and, more importantly, recovers a large amount of the FHs in the data.

Our results also show that our model is a clear improvement over the structural alerts currently in use, particularly w.r.t. dark chemical matter, on which we have significantly better results.

Finally, using a model that is not a black box allows us to present an explanation for the prediction to experts, facilitating acceptance and adding to scientific knowledge. The graphical interface that we have developed, PREPEP, is ready for use in its current form but has some limitations. We will discuss these issues and provide with some ideas on how to improve that in the conclusion and future directions chapter, which is following next.

Conclusion and future directions . . . . . 131

Publications and prototypes . . . . . 135

Acknowledgments . . . . . 137

Bibliography . . . . . 139

Appendices . . . . . 151

# Conclusion and future directions

## Conclusion

This thesis makes contributions to the field of network science by addressing the problem of link prediction in a specific type of networks which is bipartite and multi-layer at the same time. This problem has been addressed for data of biological origin that can be represented by bipartite graphs. To address the common problem of sparsity that the data of this origin has, I proposed to collect information from different sources related to that data and to combine them into a single structure represented as a multi-layer network. The state of the art approaches solving link prediction in the resulting setting have some limitations regarding the type and the numbers of layers, i.e. two similarity layers of each of bipartite types and the bipartite layer, and I had to develop my own approaches to overcome these limitations. In parallel, I tried to answer the other questions raised in the introduction: two related to link prediction and two others about the application side. In the following, I recall these questions and provide the answers, also details of the choices I made while developing the solutions.

#### Q1: Is it possible to develop an approach for link prediction in bipartite multi-layer networks without the limitations of the state of the art?

In Chapter 4, I addressed the problem of link prediction in bipartite multi-layer graphs without limiting the type and the number of layers for the first time and developed the EXPLICIT RANDOM WALKER algorithm to solve it. Based on the random walk model, the algorithm is straightforward to implement, but this simplicity results in high complexity. The random process involved in the model requires to repeat each walk many times to have acceptable results, slowing down the overall running time of the approach. The algorithm was thus kept as a baseline. Later on, in Chapter 5, I developed the NEWERMINE algorithm, which is much more efficient than the previous. In NEWERMINE, intra- and inter-network transitions are represented as matrix formulations allowing to preserve all information about all the layers. Due to the matrix multiplications involved in the new algorithm, random walk that performed in parallel in a fashion of PageRank and the absence of randomization made NEWERMINE deterministic, excluding the need to repeat the process multiple times, both of which significantly sped up the computations. The results of the evaluation presented in Chapter 6 showed that NEWERMINE also provides better prediction results than the baseline.

In Chapter 8, I developed the LPBYCD approach, which extends existing link prediction measures using community information to the case of bipartite multi-layer networks, the problem setting of this thesis. Compared to NEWERMINE, the newer approach limits the number of bipartite layers to one, which is not a serious limitation taking into account the availability of data on drug-target interaction prediction. In addition, LPBYCD has some parameters to fix before running prediction. However, those limitations are compensated by improved running time and the ability of the newest approach to auto tune its parameters. The extensive evaluation of the LPBYCD approach in Chapter 9 showed that it provides better results than NEWERMINE. So much better that they became competitive with the state of the art. I also demonstrated that the approach is indeed able to tune its parameters and that the optimal parameter setting transfers to unseen data. Moreover, I demonstrated better running time of LPBYCD compared to NEWERMINE. Finally, I assessed the scalability of the approach and demonstrated that the state of the art approaches do not scale as well. In addition to that, the state of the art approaches do not have the same level of interpretability that LPBYCD has.

**Q2: Does adding networks actually help to improve the quality of drug-target activity prediction?**

In Chapter 2, I constructed the IUPHAR network consisting of 6 layers, aimed to be used to help answering this question. In Chapter 6, I demonstrated experimentally on the full and artificially sparsified IUPHAR network that adding networks indeed helps to improve the quality of interaction predictions, positively answering the given question.

In Chapter 7, I investigated in more detail why and when adding networks helps to improve the quality of predictions. According to the results, adding networks reduces sparsity and the number of connected components, which in turn results in better vertex reachability, and thus better prediction results. But this is not always the case. At the same time, the network combinations with the minimum values of the radius, diameter and average distance correlate with optimal results in most of my experiments. As a conclusion, I would recommend to minimize these characteristics when adding new layers to the bipartite network. In addition, I draw a conclusion that the five benchmark data sets that are being widely used in the state of the art on drug-target interaction prediction are outdated due to their small sizes and connectivity characteristics they have. As an alternative, I propose to use the IUPHAR database as a more challenging benchmark instead of those sets.

**Q3: Is it possible to evaluate predicted interactions "in silico", for instance on an external knowledge source?**

Taking into account that labeled data on drug-target interaction predictions are rare and real life experiments are expensive, I made an attempt to positively answer this question. In Chapter 3, I designed a framework for validation of predicted drug-target interactions founded on an external knowledge base. I used the UMLS, a collection of biomedical concepts and their relations, as such a knowledge base. In Chapter 6, I used this framework to perform an evaluation of the interactions predicted by NEWERMINE on the IUPHAR network using the full data, i.e. without splitting it into train and test to predict new unseen interactions. The evaluation demonstrated that the results provided by the framework using new unlabeled predictions are comparable with ones we received using cross validation on labeled data. Following tests with the LPBYCD approach demonstrated one more time that the framework can be effectively used to verify predictions performed on new unlabeled data, and that optimal parameter setting can be transferred to that new data.

**Q4: Are the link prediction approaches that I develop only working with biological data?**

In Chapter 2, in addition to the IUPHAR network, I constructed generic data sets, the MovieLens and the Unicode languages networks, intended to answer this question. In Chapter 9, tests performed with the LPBYCD approach on those data sets positively answered the question. In addition, the sets were tested with other state of the art approaches and demonstrated to be challenging enough

to be new benchmarks in the field.

**Q5: Is it possible to identify and characterize promiscuous compounds using machine learning and data mining to be able to exclude them from the drug development process at an early stage?**

In Chapter 10, I worked on answering this question and proposed a solution to this problem. I developed an approach, using a mix of graph mining, sampling and machine learning for binary classification, which is able to perform detection of those compounds for the following elimination by the chemical expert. The approach exploits information from AlphaScreen assays and uses a threshold from the state of the art to discriminate between frequent and non-frequent hitters, assuming frequent hitters to be promiscuous compounds. In the experiments, I demonstrated that my approach provides better results than widely used structural alerts from the state of the art. Another advantage of the approach is its possibilities for interpretability. To demonstrate it, I developed a prototype of a graphical interface, PREPEP, providing feedback on the results for chemical experts. The prototype is ready for use, but has some limitations, which I will discuss in the next section.

## Future directions

Although contributions have been made to solving the problem of link prediction in bipartite multi-layer networks and to other research question related to an application side of the given problem in this thesis, much potential future work is still possible. In the following, I discuss these perspectives in relation to the solutions developed to the research questions recalled in the previous section.

**The NEWERMINE algorithm** It would be interesting to try NEWERMINE with biological data in another configuration. First, it would be interesting to integrate different drug-target activity databases into the multi-layer network representation. This feature provided only by NEWERMINE has not been tested yet, because of the lack of those data or the lack of expert knowledge for their construction. For future work, I would also recommend to try to integrate different drug similarity networks. In the IUPHAR network that I constructed, I have only embedded two target similarity networks, already achieving good results. Additionally, it might be interesting to employ NEWERMINE for different target settings, e.g. for miRNA (Micro Ribonucleic acid) disease links. Finally, I would try to move from the "active"/"inactive" setting to more challenging one where the *strength* of the activity is predicted.

**The LPBYCD approach** In Chapter 9, while evaluating LPBYCD, I have limited myself to two easy-to-use and less complex community detection methods. Taking into account that my approach is not dependent on the community detection method, i.e. any algorithm for solving that could be theoretically used, I would recommend to evaluate the use of other methods in the future. In addition, as I proposed for future work for NEWERMINE, it would be interesting to predict the strength of the interaction in the LPBYCD approach as well. Finally, I would recommend to try to add layers derived from other information sources to the networks from any origin and to use LPBYCD to identify possible redundancies among them in the fashion I performed with NEWERMINE while evaluating different combinations of layers. But to perform that it would probably require to move from the flattening group of community detection approaches to the direct methods to get more accurate results with different combinations of networks.

**The UMLS validation framework** In Chapter 3, I developed a framework for the validation of predicted drug-target interactions founded on an external resource, and I used the UMLS as such a resource. The experimental evaluation demonstrated that the results provided be the framework align with ones using known labels. As a next step, real biological experiments with a biochemical

expert should be carried out, to double check this point. As an intermediate step of this, interactions verified by the framework could be verified "in silico" again using an approach completely different from any discussed in this thesis, for instance by mining the corpora of biomedical articles. Finally, the choice has been made in favor of the UMLS since it was the most reliable base which I could find. The UMLS is the only knowledge base that is regularly and professionally curated by an academic institution. However, there are other knowledge bases curated by users, such as BioPortal[3] or The OBO Foundry[4]. It would be interesting to adapt my framework to one of those bases and to compare the results.

**The PREPEP prototype** As mentioned in the conclusion, there is a room for improvement for the approach and the tool I developed in Chapter 10. First of all, the tool's predictive power might be limited by the data set evaluated on one type of the assay when it comes to compounds that show frequent activity in other kinds of assays. This is not a problem per se since the prototype could be extended with subgraphs and predictive models derived from data coming from other types of bioassays. In that case, the tool would contain several predictive modules, each of which can be used to predict whether a compound is frequent hitter or not, depending on the assay type the expert is interested in, with the option to perform a consensus prediction using all modules. Related to this question is how one defines frequent hitters in the first place. In [9], any compound showing activity in two or more out of six assays was considered a frequent hitter. In their critique, [32] adopted the same threshold, and I therefore as well. It is not clear that this is an appropriate definition, however, and characterizing frequent hitters in terms of their activity percentage might be more informative. Next, the parameters I have chosen for the prototype – top-100 subgraphs, three percent minimum size for decision tree leaves, two hundred samples – were selected as a result of the trade-off of running times and performance on the data set, which I used to learn the model, but it is not clear that they are the best choices in general terms. I would recommend additional experiments on new data to verify that. Finally, all the evaluations so far have been performed entirely "in silico". To support the results, experimental biological assays are necessary.

The tool's prototype, PREPEP, has already been improved by a research engineer involved in the further development of the tool. The tool became more modular meaning that the mining, the predictive components and the graphical interface were separated. One option that has been explored but not yet implemented consists of giving experts feedback options, e.g. rejecting predictions or supposedly discriminative subgraphs. To take this feedback into account, PREPEP should become much more reactive, moving away from the off-line mining and learning that it currently performs. The component separation is a first step towards this direction, and the next is to move mining and learning modules to high-performance servers. Once it can be implemented, it would be possible to launch a web service in the future.

## General future direction

Our general future direction is rather short. I would recommend to continue research on multi-layer networks and I can explain why.

This thesis demonstrates that multi-layer networks are not fully studied yet. There were clear room for improvement for link prediction in bipartite multi-layer networks before I started my work. As I indicated in Chapter 8, there is no direct method for community detection on a multi-layer network yet which can be used out of the box. Finally, as I mentioned several times, there is a certain lack of multi-layer data. In addition to the thesis, my side research on graph embedding says that multi-layer graphs is rather challenging setting. From all of this I can only draw a conclusion that multi-layer networks are underrated. I suggest to fulfill this shortcoming in the future.

[3] `https://bioportal.bioontology.org`

[4] `http://www.obofoundry.org`

# Publications and prototypes

## Publications

The contributions presented in this thesis appear in the following publications:

### International conferences:

- Maksim Koptelov, Albrecht Zimmermann, Pascal Bonnet, Ronan Bureau and Bruno Crémilleux. "PrePeP – A Tool for the Identification and Characterization of Pan Assay Interference Compounds". *KDD'18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.
- Maksim Koptelov, Albrecht Zimmermann and Bruno Crémilleux. "Link Prediction in Multi-Layer Networks and its Application to Drug Design". *IDA'18: The Seventeenth International Symposium on Intelligent Data Analysis*, 2018.
- Maksim Koptelov, Albrecht Zimmermann, Bruno Crémilleux and Lina Soualmia. "Link Prediction via Community Detection in Bipartite Multi-Layer Graphs". *SAC'20: The 35th ACM/SIGAPP Symposium On Applied Computing*, 2020.
- Christophe Couronne, Maksim Koptelov and Albrecht Zimmermann. "PrePeP: A light-weight, extensible tool for predicting frequent hitters". Demo paper at *ECML-PKDD'20: The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.

### International workshops:

- Maksim Koptelov, Albrecht Zimmermann and Bruno Crémilleux. "Link Prediction via Community Detection in Bipartite Multi-Layer Graphs". *GEM'19: Graph Embedding and Mining workshop* as part of ECML-PKDD'19, 2019.
- Maksim Koptelov and Albrecht Zimmermann. "What can connectivity characteristics of networks tell us about the quality of link predictions?". *GEM'19: Graph Embedding and Mining workshop* as part of ECML-PKDD'19, 2019.

### Other publications:

- Maksim Koptelov, Albrecht Zimmermann, Bruno Crémilleux and Bertrand Cuissart. "Towards a structural characterization of Pan Assay Interference Compounds". Poster abstract for *SFCi'17: 8es journées de la Société Française de Chémoinformatique*, 2017.

## Prototypes

While working on this thesis, I developed the following prototypes:

**The NEWERMINE algorithm.** This algorithm is designed for the task of link prediction in bipartite multi-layer networks. The key advantage of the algorithm is that the type and the number of layers of an input network are not limited. Intra- and inter-network transitions modeled by the algorithm allow to preserve all information about the layers. The algorithm is presented in detail in Chapter 5.

- `https://github.com/koptelovmax/newermine`

**The LPBYCD approach.** This approach is designed for the same task of link prediction in bipartite multi-layer networks. The approach exploits community information of an input network to derive new links. To perform that, existing link prediction measures using community structure have been adopted for the given setting. In this approach, the type and the number of layers of an input network are not limited, except of the number of bipartite layers, which is limited to one. The key advantages of the approach are that it scales, it is able to auto-tune its parameters, also the approach has straightforward option for the result interpretability. Chapter 8 details this approach.

- `https://github.com/koptelovmax/LPbyCD`

**The UMLS validation framework.** This framework is developed for the validation of predicted drug-target interactions founded on an external resource. New unseen interactions are predicted using any drug-target interaction prediction approach, and the framework is used to assess their probability. The framework is based on the UMLS, collection of biomedical concepts curated by the National Institutes of Health. In this framework, several semantic similarity measures have been implemented and the confidence scores based on them have been introduced. The framework is described in detail in Chapter 3.

- `https://github.com/koptelovmax/umls-framework`

**The PREPEP tool.** This tool is developed to solve the problem of identification and characterization of Pan Assay Promiscuous Compounds, which exist in modern drug development. The approach inside the tool is based on graph mining, sampling and machine learning techniques. The approach exploits information from AlphaScreen assays and uses a threshold from the state of the art to discriminate between frequent and non-frequent hitters, assuming non-frequent hitters to be the promiscuous compounds. The graphical interface which the tool has is aimed to demonstrate the intrepretability of the approach. The interface is able to provide the feedback on the results for biochemical experts by illustrating the compounds which lead to the certain result of prediction. This tool is presented in Chapter 10.

- `https://github.com/koptelovmax/prepep`

# Acknowledgments

First of all, I would like to thank Christel Vrain and Tijl De Bie for accepting to review my PhD thesis. I thank them both for finding the time for this work, especially Tijl, who was chairing a major European conference. I separately thank Christel for the interest in my work and for several discussions which we had during my PhD.

I deeply thank Céline Rouveirol for accepting to be the president of the committee. I especially thank Céline for finding a way to travel to Caen within pandemic restrictions to be present in person on the defense. I really appreciate that! I deeply thank as well Erwin Bakker, who I know since my time spent in Leiden during my master. First, I thank Erwin for being all these years an external member of the thesis monitoring committee, and second, for using a vacation matter to travel from another country to Caen for my defense. I really appreciate that too!

I would like to express my very sincere thanks to my supervisors, Bruno Crémilleux, Albrecht Zimmermann and Lina Soualmia. I thank Bruno and Albrecht for believing in me and offering this PhD position at the end of my master thesis, which was also performed under their supervision. I thank you for your uncountable time and energy spent, for all the discussions we had scientific and not only, immediate response to any of my queries at non-working hours and on the weekends. I learned a lot while I was working with you! It was also a good experience and pleasant time spent together while traveling and participating in the conferences in the time when they took place off-line. I separately thank Albrecht for spending hours on correcting my writings. I learned not only on the scientific part, but also improved my writing skills while working under your supervision. I thank Lina for her patience and trust, also time spend on the numerous phone call meetings which we had. Without her involvement this thesis as a collaboration project between two research institutions in Caen and Rouen would not be possible.

I warmly thank Bertrand Cuissart for taking active part in guiding my work, especially at the beginning, and for all nice non-scientific discussions which we had. I would also like to thank Ronan Bureau from CERMN and Thierry Lecroq from the University of Rouen for the consultations they provided at the beginning of my work. In addition, I would like to thank Dino Ienco from the University of Montpellier for one important discussion which we had.

I thank the Normandy region for financing the full 3 years of this thesis. I want to thank the GREYC laboratory for provided facilities and the sysadmin team in particular for trying their best in making our work comfortable and not boring at the same time. Special thanks goes to Theo Grente, who was my office mate all this time and helped me out with translation of some parts

of this manuscript into French. I want also thank the OPTIC association for all nice events they organized for doctoral students.

This thesis would not be possible without a number of people from Jean Monnet University in Saint-Étienne. More than 5 years ago they believed in me and accepted for studying in Machine Learning and Data Mining master's program, also granted a scholarship. I should be grateful to people such as Marc Sebban, Élisa Fromont, Thomas Chaumont and some others who were involved in making this decision. I should also thank a number of professors working in the computer science department of Leiden University, where I spent an exchange semester on the second year of my master. In particular, I would like to thank Frank Takes, whose course Social Network Analysis for Computer Scientists inspired me to do PhD on a similar topic and to use networks in this thesis.

I greatly thank my numerous friends for their invaluable support: the main Russian team including Anton Z., Pavel K., Kirill and Gleb; the Primavera team, with whom I regularly traveled to music festivals and hope will continue in the future. Special thanks goes to Anton V. for being available on line at any time and supporting me in many different ways, and for summer bike trips in particular, they were real adventures! Special thanks also goes to another Anton, Dr. Bogomolov, whose example of having fun while doing PhD inspired me to repeat that experience (in my case it was not that fun, but more scientifically productive ;) ). Another special thanks goes to Pavel G., who hosted me several times while I was traveling across the Europe. Finally, I would like to thank Sergey Voronin, whose experience of pursuing education abroad motivated me to spend years mastering English and to repeat his story of immigration. Without sharing that experience this thesis probably would not exist.

I send my greetings to the guys who I know since my studies in Saint-Étienne, especially to ones I met up with multiple times in the last years: Soroush, Anastasia, Kevin, Hang, Sara. I also send my greetings to Xie and Lemon from Ghent University, who I know from participation in same scientific events and who warmly welcomed me in Ghent once.

I would also like to thank Ryan Webster, who was my best friend in Caen. Thank you Ryan for always being around and cheering me up every single day during the most tough period of the thesis writing and the confinement. I would never forget our movie nights we had in those days! Last but not least, I thank J. for supporting me all this time that I know you.

I want to finish these acknowledgments with the gratitude to my family. First, I thank my father for his continuous help and support. I thank my old grandma for raising me and always taking care of me. I thank my sister just because I have you. Finally, I dedicate this thesis to my mother who passed away before I started PhD.

# Bibliography


[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. “Link communities reveal multiscale complexity in networks”. In: *Nature* 466.7307 (2010), page 761 (cited on page 84).

[2] David Aldous and James Fill. *Reversible Markov chains and random walks on graphs*. 1995 (cited on page 45).

[3] Courtney Aldrich *et al. The ecstasy and agony of assay interference compounds*. 2017 (cited on page 118).

[4] Romas Aleliunas *et al.* “Random walks, universal traversal sequences, and the complexity of maze problems”. In: *20th Annual Symposium on Foundations of Computer Science (SFCS 1979)*. IEEE. 1979, pages 218–223 (cited on page 45).

[5] Vinicius M Alves *et al.* “Predicting chemically-induced skin reactions. Part I: QSAR models of skin sensitization and their application to identify potentially hazardous compounds”. In: *Toxicology and applied pharmacology* 284.2 (2015), pages 262–272 (cited on page 17).

[6] W Frank An and Nicola Tolliday. “Cell-based assays for high-throughput screening”. In: *Molecular biotechnology* 45.2 (2010), pages 180–186 (cited on page 2).

[7] Lars Backstrom and Jure Leskovec. “Supervised random walks: predicting and recommending links in social networks”. In: *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM. 2011, pages 635–644 (cited on page 52).

[8] Jonathan B Baell. “Feeling nature’s PAINS: Natural products, natural product drugs, and pan assay interference compounds (PAINS)”. In: *Journal of natural products* 79.3 (2016), pages 616–628 (cited on page 120).

[9] Jonathan B Baell and Georgina A Holloway. “New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays”. In: *Journal of medicinal chemistry* 53.7 (2010), pages 2719–2740 (cited on pages 2, 118, 120, 123, 124, 134).

[10] Jonathan Baell and Michael A Walters. “Chemistry: Chemical con artists foil drug discovery”. In: *Nature News* 513.7519 (2014), page 481 (cited on page 120).

[11] Jürgen Bajorath. "Pharmacophore". In: *Encyclopedia of Cancer*. Edited by Manfred Schwab. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pages 1–4. ISBN: 978-3-642-27841-9. DOI: 10.1007/978-3-642-27841-9_4502-2. URL: https://doi.org/10.1007/978-3-642-27841-9_4502-2 (cited on page 8).

[12] Albert-László Barabási *et al. Network science*. Cambridge university press, 2016 (cited on pages 14, 88, 89).

[13] Bcary. *Pan-assay interference compounds*. 2016. URL: https://commons.wikimedia.org/wiki/File:PAINS_Figure.tif (cited on page 119).

[14] Andreas Bender *et al.* "Molecular similarity searching using atom environments, information-based feature selection, and a naive Bayesian classifier". In: *Journal of chemical information and computer sciences* 44.1 (2004), pages 170–178 (cited on page 17).

[15] Michael A Bender and Donna K Slonim. "The power of team exploration: Two robots can learn unlabeled directed graphs". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE. 1994, pages 75–85 (cited on page 45).

[16] M. R. Berthold *et al.* "KNIME: The Konstanz Information Miner". In: Springer, 2007. Chapter Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007) (cited on page 123).

[17] BIOVIA. *BIOVIA Pipeline Pilot*. 2017. URL: http://accelrys.com/products/collaborative-science/biovia-pipeline-pilot/ (cited on page 123).

[18] BIOVIA. *CTfile Formats*. 2017. URL: http://accelrys.com/products/collaborative-science/biovia-draw/ctfile-no-fee.html (cited on page 127).

[19] Kevin Bleakley and Yoshihiro Yamanishi. "Supervised prediction of drug–target interactions using bipartite local models". In: *Bioinformatics* 25.18 (2009), pages 2397–2403 (cited on pages 17, 92).

[20] Vincent D Blondel *et al.* "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008 (cited on pages 85, 88).

[21] Olivier Bodenreider. "The unified medical language system (UMLS): integrating biomedical terminology". In: *Nucleic acids research* 32.suppl_1 (2004), pages D267–D270 (cited on pages 31–33).

[22] Olivier Bodenreider and Alexa T McCray. "Exploring semantic groups through visual approaches". In: *Journal of biomedical informatics* 36.6 (2003), pages 414–432 (cited on page 33).

[23] Nicolas Bosc *et al.* "Prediction of protein kinase–ligand interactions through 2.5 D kinochemometrics". In: *Journal of chemical information and modeling* 57.1 (2017), pages 93–101 (cited on page 123).

[24] Markus Bredel and Edgar Jacoby. "Chemogenomics: an emerging strategy for rapid target and drug discovery". In: *Nature Reviews Genetics* 5.4 (2004), pages 262–275 (cited on pages 8, 17).

[25] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pages 123–140 (cited on page 122).

[26] Leo Breiman *et al. Classification and Regression Trees*. New York: Chapman & Hall, 1984, page 358. ISBN: 0-412-04841-8 (cited on page 124).

[27] Björn Bringmann *et al.* “Don’t be afraid of simpler patterns”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2006, pages 55–66 (cited on pages 119–121).

[28] Krisztian Buza and Ladislav Peska. “ALADIN: A New Approach for Drug–Target Interaction Prediction”. In: *ECML/PKDD*. Springer. 2017, pages 322–337 (cited on pages 2, 18, 24, 92, 106, 110, 111).

[29] Tolga Can, Orhan Çamoğlu, and Ambuj K Singh. “Analysis of protein-protein interaction networks using random walks”. In: *Proceedings of the 5th international workshop on Bioinformatics*. ACM. 2005, pages 61–68 (cited on pages 45, 46).

[30] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. “From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks”. In: *Scientific reports* 3 (2013), page 1613 (cited on pages 84, 86).

[31] Dong-Sheng Cao *et al.* “Computational Prediction of Drug–Target Interactions Using Chemical, Biological, and Network Features”. In: *Molecular informatics* 33.10 (2014), pages 669–681 (cited on page 18).

[32] Stephen J Capuzzi, Eugene N Muratov, and Alexander Tropsha. “Phantom PAINS: Problems with the Utility of Alerts for Pan-Assay INterference CompoundS”. In: *Journal of chemical information and modeling* 57.3 (2017), pages 417–427 (cited on pages 118, 120, 123, 134).

[33] Lei Chen and Wei-Ming Zeng. “A two-step similarity-based method for prediction of drug’s target group”. In: *Protein and peptide letters* 20.3 (2013), pages 364–370 (cited on page 17).

[34] Xing Chen, Ming-Xi Liu, and Gui-Ying Yan. “Drug–target interaction prediction by random walk on the heterogeneous network”. In: *Molecular BioSystems* 8.7 (2012), pages 1970–1978 (cited on pages 2, 18, 24, 52, 55, 65).

[35] Xing Chen *et al.* “Drug–target interaction prediction: databases, web servers and computational models”. In: *Briefings in bioinformatics* 17.4 (2016), pages 696–712 (cited on page 18).

[36] Alan C Cheng *et al.* “Structure-based maximal affinity model predicts small-molecule druggability”. In: *Nature biotechnology* 25.1 (2007), pages 71–75 (cited on page 17).

[37] Feixiong Cheng *et al.* “Prediction of chemical-protein interactions network with weighted network-based inference method”. In: *PloS One* 7.7 (2012), e41064 (cited on page 18).

[38] Tiejun Cheng *et al.* “Identifying compound-target associations by combining bioactivity profile similarity search and public databases mining”. In: *Journal of chemical information and modeling* 51.9 (2011), pages 2440–2448 (cited on page 17).

[39] Tiejun Cheng *et al.* “Structure-based virtual screening for drug discovery: a problem-centric review”. In: *The AAPS journal* 14.1 (2012), pages 133–141 (cited on page 17).

[40] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. “Hierarchical structure and the prediction of missing links in networks”. In: *Nature* 453.7191 (2008), page 98 (cited on page 83).

[41] Don Coppersmith, Uriel Feige, and James Shearer. “Random walks on regular and irregular graphs”. In: *SIAM Journal on Discrete Mathematics* 9.2 (1996), pages 301–308 (cited on page 45).

[42] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pages 273–297 (cited on page 122).

[43] John G Cumming *et al.* "Chemical predictive modelling to improve compound quality". In: *Nature reviews Drug discovery* 12.12 (2013), page 948 (cited on page 120).

[44] Jayme L Dahlin *et al.* "PAINS in the assay: chemical mechanisms of assay interference and promiscuous enzymatic inhibition observed during a sulfhydryl-scavenging HTS". In: *Journal of medicinal chemistry* 58.5 (2015), pages 2091–2113 (cited on page 120).

[45] Fred J Damerau. "A technique for computer detection and correction of spelling errors". In: *Communications of the ACM* 7.3 (1964), pages 171–176 (cited on page 26).

[46] Mindy I Davis *et al.* "Comprehensive analysis of kinase inhibitor selectivity". In: *Nature biotechnology* 29.11 (2011), page 1046 (cited on pages 24, 25).

[47] Caterina De Bacco *et al.* "Community detection, link prediction, and layer interdependence in multilayer networks". In: *Physical Review E* 95.4 (2017), page 042317 (cited on page 84).

[48] Dina Demner-Fushman, Willie J Rogers, and Alan R Aronson. "MetaMap Lite: an evaluation of a new Java implementation of MetaMap". In: *Journal of the American Medical Informatics Association* 24.4 (2017), pages 841–844 (cited on page 32).

[49] Hao Ding *et al.* "Similarity-based machine learning methods for predicting drug–target interactions: a brief review". In: *Briefings in bioinformatics* 15.5 (2013), pages 734–747 (cited on page 18).

[50] Jingyi Ding *et al.* "Prediction of missing links based on community relevance and ruler inference". In: *Knowledge-Based Systems* 98 (2016), pages 200–215 (cited on pages 84, 86).

[51] Oranit Dror *et al.* "Novel approach for efficient pharmacophore-based virtual screening: method and applications". In: *Journal of chemical information and modeling* 49.10 (2009), pages 2333–2343 (cited on page 8).

[52] Richard M Eglen *et al.* "The use of AlphaScreen technology in HTS: current status". In: *Current chemical genomics* 1 (2008), page 2 (cited on page 126).

[53] Ali Ezzat *et al.* "Computational prediction of drug–target interactions using chemogenomic approaches: an empirical survey". In: *Briefings in bioinformatics* 20.4 (2019), pages 1337–1357 (cited on page 17).

[54] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pages 861–874 (cited on page 16).

[55] Miroslav Fiedler. "Algebraic connectivity of graphs". In: *Czechoslovak mathematical journal* 23.2 (1973), pages 298–305 (cited on page 87).

[56] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pages 75–174 (cited on pages 87, 88).

[57] Valerio Freschi. "Protein function prediction from interaction networks using a random walk ranking algorithm". In: *2007 IEEE 7th International Symposium on BioInformatics and BioEngineering*. IEEE. 2007, pages 42–48 (cited on page 52).

[58] Jean Gallier. "Notes on elementary spectral graph theory. Applications to graph clustering using normalized cuts". In: *arXiv preprint arXiv:1311.2492* (2013) (cited on page 12).

[59] José-Manuel Gally *et al.* "VSPrep: a general KNIME workflow for the Preparation of molecules for virtual screening". In: *Molecular informatics* 36.10 (2017), page 1700023 (cited on page 123).

[60] Vijay N Garla and Cynthia Brandt. "Semantic similarity in the biomedical domain: an evaluation across knowledge sources". In: *BMC bioinformatics* 13.1 (2012), page 261 (cited on page 34).

[61] Michelle Girvan and Mark EJ Newman. “Community structure in social and biological networks”. In: *Proceedings of the national academy of sciences* 99.12 (2002), pages 7821–7826 (cited on page 15).

[62] F Göbel and AA Jagers. “Random walks on graphs”. In: *Stochastic processes and their applications* 2.4 (1974), pages 311–336 (cited on page 45).

[63] Kwang-Il Goh *et al.* “The human disease network”. In: *Proceedings of the National Academy of Sciences* 104.21 (2007), pages 8685–8690 (cited on page 15).

[64] Mehmet Gönen. “Predicting drug–target interactions from chemical and genomic kernels using Bayesian matrix factorization”. In: *Bioinformatics* 28.18 (2012), pages 2304–2310 (cited on page 18).

[65] Swati Goswami, CA Murthy, and Asit K Das. “Sparsity measure of a network graph: Gini index”. In: *Information Sciences* 462 (2018), pages 16–39 (cited on page 11).

[66] Palash Goyal and Emilio Ferrara. “Graph embedding techniques, applications, and performance: A survey”. In: *Knowledge-Based Systems* 151 (2018), pages 78–94 (cited on page 18).

[67] Stephen Guattery and Gary L Miller. “On the performance of spectral graph partitioning methods”. In: *SODA*. Volume 95. 1995, pages 233–242 (cited on page 87).

[68] Rajarshi Guha. “On the interpretation and interpretability of quantitative structure–activity relationship models”. In: *Journal of computer-aided molecular design* 22.12 (2008), pages 857–871 (cited on page 120).

[69] Roger Guimerà and Marta Sales-Pardo. “Missing and spurious interactions and the reconstruction of complex networks”. In: *Proceedings of the National Academy of Sciences* 106.52 (2009), pages 22073–22078 (cited on pages 83, 84).

[70] Stefan Günther *et al.* “SuperTarget and Matador: resources for exploring drug-target relationships”. In: *Nucleic acids research* 36.suppl_1 (2007), pages D919–D922 (cited on pages 24, 72).

[71] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Technical report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008 (cited on page 124).

[72] F Maxwell Harper and Joseph A Konstan. “The movielens datasets: History and context”. In: *ACM transactions on interactive intelligent systems (tiis)* 5.4 (2015), pages 1–19 (cited on page 25).

[73] Masahiro Hattori *et al.* “Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways”. In: *Journal of the American Chemical Society* 125.39 (2003), pages 11853–11865 (cited on page 24).

[74] Taher H Haveliwala. “Topic-sensitive pagerank”. In: *Proceedings of the 11th international conference on World Wide Web*. ACM. 2002, pages 517–526 (cited on page 52).

[75] Taher H Haveliwala. “Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search”. In: *IEEE transactions on knowledge and data engineering* 15.4 (2003), pages 784–796 (cited on page 54).

[76] Zhisong He *et al.* “Predicting drug-target interaction networks based on functional groups and biological features”. In: *PloS One* 5.3 (2010) (cited on page 18).

[77] Kazi Yasin Helal *et al.* "Public domain HTS fingerprints: design and evaluation of compound bioactivity profiles from PubChem's bioassay repository". In: *Journal of chemical information and modeling* 56.2 (2016), pages 390–398 (cited on page 17).

[78] Jonathan L. Herlocker *et al.* "An algorithmic framework for performing collaborative filtering". In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*. Association for Computing Machinery, Inc, Aug. 1999, pages 230–237 (cited on page 25).

[79] Desislava Hristova *et al.* "A multilayer approach to multiplexity and link prediction in online geo-social networks". In: *EPJ Data Science* 5.1 (2016), page 24 (cited on page 84).

[80] Daylight Chemical Information Systems Inc. *Simplified Molecular Input Line Entry System*. URL: `http://www.daylight.com/smiles/index.html` (cited on page 123).

[81] Akihiro Inokuchi and Takashi Washio. "A fast method to mine frequent subsequences from graph sequence data". In: *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE. 2008, pages 303–312 (cited on page 120).

[82] Mahdi Jalili *et al.* "Link prediction in multiplex online social networks". In: *Royal Society open science* 4.2 (2017), page 160863 (cited on page 84).

[83] Ali Akbar Jamali *et al.* "DrugMiner: comparative analysis of machine learning algorithms for prediction of potential druggable proteins". In: *Drug discovery today* 21.5 (2016), pages 718–724 (cited on page 18).

[84] Swarit Jasial, Ye Hu, and Jürgen Bajorath. "How frequently are pan-assay interference compounds active? Large-scale analysis of screening data reveals diverse activity profiles, low global hit frequency, and many consistently inactive compounds". In: *Journal of Medicinal Chemistry* 60.9 (2017), pages 3879–3886 (cited on pages 118, 120).

[85] Glen Jeh and Jennifer Widom. "Scaling personalized web search". In: *Proceedings of the 12th international conference on World Wide Web*. ACM. 2003, pages 271–279 (cited on page 52).

[86] Ulf Johansson *et al.* "Trade-off between accuracy and interpretability for predictive in silico modeling". In: *Future medicinal chemistry* 3.6 (2011), pages 647–663 (cited on page 120).

[87] Minoru Kanehisa *et al.* "KEGG for linking genomes to life and the environment". In: *Nucleic acids research* 36.suppl_1 (2007), pages D480–D484 (cited on pages 24, 72).

[88] Anna R Karlin and Prabhakar Raghavan. "Random walks and undirected graph connectivity: A survey". In: *Discrete Probability and Algorithms*. Springer, 1995, pages 95–101 (cited on page 46).

[89] Michael J Keiser *et al.* "Predicting new molecular targets for known drugs". In: *Nature* 462.7270 (2009), pages 175–181 (cited on page 17).

[90] Sunghwan Kim *et al.* "PubChem substance and compound databases". In: *Nucleic acids research* 44.D1 (2016), pages D1202–D1213 (cited on page 123).

[91] Mikko Kivelä *et al.* "Multilayer networks". In: *Journal of complex networks* 2.3 (2014), pages 203–271 (cited on page 11).

[92] Alexios Koutsoukas *et al.* "From in silico target prediction to multi-target drug design: current databases, methods and applications". In: *Journal of proteomics* 74.12 (2011), pages 2554–2574 (cited on page 17).

[93] Zhana Kuncheva and Giovanni Montana. “Community detection in multiplex networks using locally adaptive random walks”. In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM. 2015, pages 1308–1315 (cited on page 84).

[94] Jérôme Kunegis. “Konect: the koblenz network collection”. In: *Proceedings of the 22nd International Conference on World Wide Web*. 2013, pages 1343–1350 (cited on page 28).

[95] Michihiro Kuramochi and George Karypis. “Frequent subgraph discovery”. In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE international conference on*. IEEE. 2001, pages 313–320 (cited on page 120).

[96] Renaud Lambiotte, J-C Delvenne, and Mauricio Barahona. “Laplacian dynamics and multiscale modular structure in networks”. In: *arXiv preprint arXiv:0812.1770* (2008) (cited on page 89).

[97] H Vernon Leighton and Jaideep Srivastava. “First 20 precision among World Wide Web search services (search engines)”. In: *Journal of the American Society for Information Science* 50.10 (1999), pages 870–881 (cited on page 63).

[98] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014 (cited on pages 15, 47, 53, 85, 87).

[99] Honglin Li *et al.* “TarFisDock: a web server for identifying drug targets with docking approach”. In: *Nucleic acids research* 34.suppl_2 (2006), W219–W224 (cited on page 17).

[100] Xin Li and Hsinchun Chen. “Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach”. In: *Decision Support Systems* 54.2 (2013), pages 880–890 (cited on page 1).

[101] David Liben-Nowell and Jon Kleinberg. “The link-prediction problem for social networks”. In: *Journal of the American society for information science and technology* 58.7 (2007), pages 1019–1031 (cited on pages 84, 85).

[102] Hansaim Lim *et al.* “Improved genome-scale multi-target virtual screening via a novel collaborative filtering approach to cold-start problem”. In: *Scientific reports* 6 (2016), page 38860 (cited on pages 2, 18, 24).

[103] Dekang Lin *et al.* “An information-theoretic definition of similarity.” In: *Icml*. Volume 98. 1998. 1998, pages 296–304 (cited on page 34).

[104] Weiping Liu and Linyuan Lü. “Link prediction based on local random walk”. In: *EPL (Europhysics Letters)* 89.5 (2010), page 58007 (cited on page 52).

[105] Yong Liu *et al.* “Neighborhood regularized logistic matrix factorization for drug-target interaction prediction”. In: *PLoS computational biology* 12.2 (2016) (cited on page 18).

[106] Yuansheng Liu *et al.* “Inferring microRNA-disease associations by random walk on a heterogeneous network with multiple data sources”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 14.4 (2016), pages 905–915 (cited on page 52).

[107] Yuansheng Liu *et al.* “Inferring microRNA-disease associations by random walk on a heterogeneous network with multiple data sources”. In: *TCBB* 14.4 (2017), pages 905–915 (cited on page 52).

[108] Zhen Liu *et al.* “Correlations between community structure and link formation in complex networks”. In: *PloS One* 8.9 (2013), e72908 (cited on page 84).

[109] László Lovász *et al.* “Random walks on graphs: A survey”. In: *Combinatorics, Paul erdos is eighty* 2.1 (1993), pages 1–46 (cited on pages 45, 46).

[110] Sylvain Lozano *et al.* "Introduction of jumping fragments in combination with QSARs for the assessment of classification in ecotoxicology". In: *Journal of chemical information and modeling* 50.8 (2010), pages 1330–1339 (cited on pages 119, 120).

[111] Jiawei Luo and Qiu Xiao. "A novel approach for predicting microRNA-disease associations by unbalanced bi-random walk on heterogeneous network". In: *Journal of biomedical informatics* 66 (2017), pages 194–203 (cited on page 52).

[112] Thusitha Mabotuwana, Michael C Lee, and Eric V Cohen-Solal. "An ontology-based similarity measure for biomedical data–Application to radiology reports". In: *Journal of biomedical informatics* 46.5 (2013), pages 857–868 (cited on page 34).

[113] Matthew K Matlock *et al.* "Modeling small-molecule reactivity identifies promiscuous bioactive compounds". In: *Journal of Chemical Information and Modeling* 58.8 (2018), pages 1483–1500 (cited on page 120).

[114] Jian-Ping Mei *et al.* "Drug–target interaction prediction by learning from local information and neighbors". In: *Bioinformatics* 29.2 (2013), pages 238–245 (cited on pages 17, 110).

[115] Jörg Menche *et al.* "Uncovering disease-disease relationships through the incomplete interactome". In: *Science* 347.6224 (2015), page 1257601 (cited on page 15).

[116] Thomas Mendgen, Christian Steuer, and Christian D Klein. "Privileged scaffolds or promiscuous binders: a comparative study on rhodanines and related heterocycles in medicinal chemistry". In: *Journal of medicinal chemistry* 55.2 (2012), pages 743–753 (cited on page 120).

[117] Jean-Philippe Métivier *et al.* "Discovering structural alerts for mutagenicity using stable emerging molecular patterns". In: *Journal of chemical information and modeling* 55.5 (2015), pages 925–940 (cited on pages 119, 120).

[118] Milena Mihail and Christos H Papadimitriou. "On the random walk method for protocol testing". In: *International Conference on Computer Aided Verification*. Springer. 1994, pages 132–141 (cited on page 45).

[119] Shinichi Morishita and Jun Sese. "Transversing itemset lattices with statistical metric pruning". In: *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2000, pages 226–236 (cited on page 120).

[120] Garrett M Morris *et al.* "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility". In: *Journal of computational chemistry* 30.16 (2009), pages 2785–2791 (cited on page 17).

[121] Mark A Musen *et al.* "The national center for biomedical ontology". In: *Journal of the American Medical Informatics Association* 19.2 (2012), pages 190–195 (cited on page 31).

[122] Kathryn M Nelson *et al.* "The essential medicinal chemistry of curcumin: miniperspective". In: *Journal of medicinal chemistry* 60.5 (2017), pages 1620–1637 (cited on pages 118, 120).

[123] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical review E* 69.2 (2004), page 026113 (cited on page 88).

[124] Nidhi *et al.* "Prediction of biological targets for compounds using multiple-category Bayesian models trained on chemogenomics databases". In: *Journal of chemical information and modeling* 46.3 (2006), pages 1124–1133 (cited on page 17).

[125] Siegfried Nijssen and Joost Kok. "Frequent subgraph miners: runtimes don't say everything". In: *Proceedings of the Workshop on Mining and Learning with Graphs*. 2006, pages 173–180 (cited on page 120).

[126] Siegfried Nijssen and Joost N Kok. “A quickstart in frequent structure mining can make a difference”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pages 647–652 (cited on page 120).

[127] Noel M O’Boyle *et al.* “Open Babel: An open chemical toolbox”. In: *Journal of cheminformatics* 3.1 (2011), page 33 (cited on page 127).

[128] Lawrence Page *et al. The PageRank citation ranking: Bringing order to the web*. Technical report. Stanford InfoLab, 1999 (cited on page 51).

[129] Adam J Pawson *et al.* “The IUPHAR/BPS Guide to PHARMACOLOGY: an expert-driven knowledgebase of drug targets and their ligands”. In: *Nucleic acids research* 42.D1 (2014), pages D1098–D1106 (cited on page 21).

[130] Karl Pearson. “The problem of the random walk”. In: *Nature* 72.1867 (1905), pages 342–342 (cited on page 45).

[131] Fabian Pedregosa *et al.* “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pages 2825–2830 (cited on page 124).

[132] Dariusz Plewczynski *et al.* “Target specific compound identification using a support vector machine”. In: *Combinatorial chemistry & high throughput screening* 10.3 (2007), pages 189–196 (cited on page 17).

[133] Martin Pouliot and Stephane Jeanmart. “Pan Assay Interference Compounds (PAINS) and Other Promiscuous Compounds in Antifungal Research: Miniperspective”. In: *Journal of medicinal chemistry* 59.2 (2015), pages 497–503 (cited on pages 118, 120).

[134] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. “NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins”. In: *Nucleic acids research* 33.suppl_1 (2005), pages D501–D504 (cited on page 23).

[135] Gerard Pujadas *et al.* “Protein-ligand docking: A review of recent advances and future perspectives”. In: *Current Pharmaceutical Analysis* 4.1 (2008), pages 1–19 (cited on page 17).

[136] Roy Rada *et al.* “Development and application of a metric on semantic nets”. In: *IEEE transactions on systems, man, and cybernetics* 19.1 (1989), pages 17–30 (cited on page 34).

[137] Erzsébet Ravasz *et al.* “Hierarchical organization of modularity in metabolic networks”. In: *science* 297.5586 (2002), pages 1551–1555 (cited on page 15).

[138] *RDKit: Open-Source Cheminformatics*. 2017. URL: `http://www.rdkit.org` (cited on page 123).

[139] Omer Reingold. “Undirected connectivity in log-space”. In: *Journal of the ACM (JACM)* 55.4 (2008), pages 1–24 (cited on page 46).

[140] Philip Resnik. “Using information content to evaluate semantic similarity in a taxonomy”. In: *arXiv preprint cmp-lg/9511007* (1995) (cited on page 35).

[141] Ulrich Rester. “From virtuality to reality-Virtual screening in lead discovery and lead optimization: a medicinal chemistry perspective.” In: *Current opinion in drug discovery & development* 11.4 (2008), pages 559–568 (cited on page 8).

[142] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pages 1135–1144 (cited on page 122).

[143] Purnamrita Sarkar. “Random Walks on Graphs: An Overview”. In: () (cited on page 53).

[144] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pages 85–117 (cited on page 122).

[145] Ida Schomburg, Antje Chang, and Dietmar Schomburg. “BRENDA, enzyme data and metabolic information”. In: *Nucleic acids research* 30.1 (2002), pages 47–49 (cited on pages 24, 72).

[146] Barry Smith *et al.* “The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration”. In: *Nature biotechnology* 25.11 (2007), pages 1251–1255 (cited on page 31).

[147] Temple F Smith, Michael S Waterman, *et al.* “Identification of common molecular subsequences”. In: *Journal of molecular biology* 147.1 (1981), pages 195–197 (cited on page 24).

[148] Sucheta Soundarajan and John Hopcroft. “Using community information to improve the precision of link prediction methods”. In: *Proceedings of the 21st International Conference on World Wide Web*. ACM. 2012, pages 607–608 (cited on page 84).

[149] Daniel A Spielman and Shang-Hua Teng. “Spectral partitioning works: Planar graphs and finite element meshes”. In: *Linear Algebra and its Applications* 421.2-3 (2007), pages 284–305 (cited on page 102).

[150] Chris Stark *et al.* “BioGRID: a general repository for interaction datasets”. In: *Nucleic acids research* 34.suppl_1 (2006), pages D535–D539 (cited on pages 23, 72).

[151] Conrad Stork *et al.* “Hit Dexter 2.0: machine-learning models for the prediction of frequent hitters”. In: *Journal of chemical information and modeling* 59.3 (2019), pages 1030–1043 (cited on page 120).

[152] Jimeng Sun *et al.* “Neighborhood formation and anomaly detection in bipartite graphs”. In: *Fifth IEEE International Conference on Data Mining (ICDM’05)*. IEEE. 2005, 8–pp (cited on page 1).

[153] S Joshua Swamidass *et al.* “Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity”. In: *Bioinformatics* 21.suppl_1 (2005), pages i359–i368 (cited on page 120).

[154] Andrea Tagarelli, Alessia Amelio, and Francesco Gullo. “Ensemble-based community detection in multilayer networks”. In: *Data Mining and Knowledge Discovery* 31.5 (2017), pages 1506–1543 (cited on pages 84, 85).

[155] SM Tasso *et al.* “Pharmacophore searching and QSAR analysis in the design of anticonvulsant drugs”. In: *Journal of molecular structure: THEOCHEM* 504.1-3 (2000), pages 229–240 (cited on page 17).

[156] Natasha Thorne, Douglas S Auld, and James Inglese. “Apparent activity in high-throughput screening: origins of compound-dependent assay interference”. In: *Current opinion in chemical biology* 14.3 (2010), pages 315–324 (cited on page 120).

[157] Tihomir Tomašić and Lucija Peterlin Mašič. “Rhodanine as a scaffold in drug discovery: a critical review of its biological activities and mechanisms of target modulation”. In: *Expert opinion on drug discovery* 7.7 (2012), pages 549–560 (cited on page 120).

[158] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. “Fast random walk with restart and its applications”. In: *Sixth International Conference on Data Mining (ICDM’06)*. IEEE. 2006, pages 613–622 (cited on page 52).

[159] Jorge Carlos Valverde-Rebaza and Alneu de Andrade Lopes. “Link prediction in online social networks using group information”. In: *International Conference on Computational Science and Its Applications*. Springer. 2014, pages 31–45 (cited on pages 83, 84).

[160] Twan Van Laarhoven and Elena Marchiori. “Predicting drug-target interactions for new drug compounds using a weighted nearest neighbor profile”. In: *PloS One* 8.6 (2013) (cited on pages 17, 110).

[161] Santiago Vilar *et al.* “Computational drug target screening through protein interaction profiles”. In: *Scientific reports* 6 (2016), page 36969 (cited on page 17).

[162] Nikil Wale and George Karypis. “Target fishing for chemical compounds using target-ligand activity data and ranking based methods”. In: *Journal of chemical information and modeling* 49.10 (2009), pages 2190–2201 (cited on page 17).

[163] David S Wishart *et al.* “DrugBank: a comprehensive resource for in silico drug discovery and exploration”. In: *Nucleic acids research* 34.suppl_1 (2006), pages D668–D672 (cited on pages 23, 72).

[164] David J Wood *et al.* “Automated QSAR with a hierarchy of global and local models”. In: *Molecular informatics* 30.11-12 (2011), pages 960–972 (cited on page 120).

[165] Marc Wörlein *et al.* “A quantitative comparison of the subgraph miners MoFa, gSpan, FFSM, and Gaston”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2005, pages 392–403 (cited on page 120).

[166] Zheng Xia *et al.* “Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces”. In: *BMC systems biology*. Volume 4. 2. BioMed Central. 2010, S6 (cited on pages 17, 110).

[167] Maoqiang Xie, Taehyun Hwang, and Rui Kuang. “Prioritizing disease genes by bi-random walk”. In: *PAKDD*. Springer. 2012, pages 292–303 (cited on page 52).

[168] Zheng Xie *et al.* “Potential links by neighbor communities”. In: *Physica A: Statistical Mechanics and its Applications* 406 (2014), pages 244–252 (cited on pages 84, 86).

[169] Wenpu Xing and Ali Ghorbani. “Weighted pagerank algorithm”. In: *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.* IEEE. 2004, pages 305–314 (cited on page 52).

[170] Yoshihiro Yamanishi *et al.* “Prediction of drug–target interaction networks from the integration of chemical and genomic spaces”. In: *Bioinformatics* 24.13 (2008), pages i232–i240 (cited on pages 17, 18, 24, 25).

[171] Yoshihiro Yamanishi *et al.* “Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework”. In: *Bioinformatics* 26.12 (2010), pages i246–i254 (cited on page 18).

[172] Bowen Yan and Steve Gregory. “Finding missing edges in networks based on their community structure”. In: *Physical Review E* 85.5 (2012), page 056112 (cited on pages 83, 84).

[173] Xifeng Yan and Jiawei Han. “gspan: Graph-based substructure pattern mining”. In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings*. IEEE. 2002, pages 721–724 (cited on pages 23, 119, 120).

[174] Jeremy J Yang *et al.* “Badapple: promiscuity patterns from noisy evidence”. In: *Journal of cheminformatics* 8.1 (2016), page 29 (cited on page 120).

[175] Sheng-Yong Yang. “Pharmacophore modeling and applications in drug discovery: challenges and recent advances”. In: *Drug discovery today* 15.11-12 (2010), pages 444–450 (cited on page 17).

[176] Zhijun Yin *et al.* "A unified framework for link recommendation using random walks". In: *2010 International Conference on Advances in Social Networks Analysis and Mining*. IEEE. 2010, pages 152–159 (cited on page 52).

[177] Hua Yu *et al.* "A systematic prediction of multiple drug-target interactions from chemical, genomic, and pharmacological data". In: *PloS One* 7.5 (2012) (cited on page 18).

[178] Xiaodong Zheng *et al.* "Collaborative matrix factorization with multiple similarities for predicting drug-target interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pages 1025–1033 (cited on pages 18, 24).

[179] Zhuojie Zhou *et al.* "Faster random walks by rewiring online social networks on-the-fly". In: *ACM Transactions on Database Systems (TODS)* 40.4 (2016), page 26 (cited on page 52).

[180] Albrecht Zimmermann, Björn Bringmann, and Ulrich Rückert. "Fast, effective molecular feature mining by local optimization". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2010, pages 563–578 (cited on pages 119, 120).

# Appendices

*There are two appendices in this thesis:*

- *Appendix 1 provides additional results for the link prediction via community detection evaluation performed in Chapter 9.*
- *Appendix 2 provides additional results for the approach for detection and characterization of promiscuous compounds evaluation performed in Chapter 10.*

## APPENDIX 1

This appendix provides additional results for the parameter selection via internal cross-validation experiment performed in Chapter 9.

Table 1: Spectral partitioning parameter optimization via internal cross-validation on benchmark data sets (Enzyme)

| Data set | Measure | Aggregation function | Fold | Internal CV | | | | External CV | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Optimal parameters | | AUC | AUPR | AUC | AUPR |
| | | | | thresholding | m | | | | |
| Enzyme | $JC_{CC}$ | – | 1 | global median | 17 | 0.85 | 0.13 | 0.87 | 0.28 |
| | | | 2 | default | 16 | 0.85 | 0.18 | 0.84 | 0.23 |
| | | | 3 | global mean | 12 | 0.85 | 0.11 | 0.85 | 0.19 |
| | | | 4 | local median | 10 | 0.85 | 0.10 | 0.85 | 0.10 |
| | | | 5 | local mean | 15 | 0.85 | 0.18 | 0.85 | 0.14 |
| | $JC_{NC}$ | max | 1 | global median | 5 | 0.91 | 0.18 | 0.92 | 0.31 |
| | | | 2 | local median | 5 | 0.91 | 0.16 | 0.90 | 0.21 |
| | | | 3 | local median | 6 | 0.91 | 0.19 | 0.90 | 0.29 |
| | | | 4 | global mean | 5 | 0.91 | 0.17 | 0.92 | 0.27 |
| | | | 5 | local median | 5 | 0.91 | 0.17 | 0.93 | 0.24 |
| | | mean | 1 | local sum | 5 | 0.91 | 0.19 | 0.93 | 0.34 |
| | | | 2 | individual sum | 3 | 0.91 | 0.17 | 0.90 | 0.19 |
| | | | 3 | local median | 6 | 0.92 | 0.22 | 0.91 | 0.29 |
| | | | 4 | local mean | 4 | 0.91 | 0.19 | 0.92 | 0.24 |
| | | | 5 | local median | 6 | 0.91 | 0.18 | 0.92 | 0.23 |

Table 2: Spectral partitioning parameter optimization via internal cross-validation on benchmark data sets (GPCR, IC, NR)

| Data set | Measure | Aggregation function | Fold | Internal CV | | | | External CV | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Optimal parameters | | AUC | AUPR | AUC | AUPR |
| | | | | thresholding | m | | | | |
| GPCR | $JC_{CC}$ | – | 1 | global mean | 12 | 0.79 | 0.11 | 0.78 | 0.14 |
| | | | 2 | local median | 13 | 0.79 | 0.14 | 0.79 | 0.13 |
| | | | 3 | local median | 8 | 0.78 | 0.10 | 0.81 | 0.17 |
| | | | 4 | local median | 15 | 0.78 | 0.11 | 0.82 | 0.18 |
| | | | 5 | global mean | 11 | 0.80 | 0.11 | 0.81 | 0.13 |
| | $JC_{NC}$ | max | 1 | global mean | 6 | 0.84 | 0.18 | 0.84 | 0.24 |
| | | | 2 | individual mean | 5 | 0.84 | 0.20 | 0.86 | 0.24 |
| | | | 3 | local sum | 4 | 0.83 | 0.18 | 0.86 | 0.23 |
| | | | 4 | global median | 4 | 0.83 | 0.19 | 0.84 | 0.23 |
| | | | 5 | individual mean | 4 | 0.84 | 0.18 | 0.85 | 0.27 |
| | | mean | 1 | local mean | 6 | 0.84 | 0.19 | 0.84 | 0.24 |
| | | | 2 | global median | 7 | 0.84 | 0.21 | 0.86 | 0.24 |
| | | | 3 | individual sum | 5 | 0.83 | 0.17 | 0.84 | 0.22 |
| | | | 4 | local sum | 4 | 0.83 | 0.20 | 0.84 | 0.26 |
| | | | 5 | local mean | 5 | 0.85 | 0.19 | 0.85 | 0.27 |
| IC | $JC_{CC}$ | – | 1 | individual median | 12 | 0.83 | 0.33 | 0.81 | 0.45 |
| | | | 2 | individual median | 11 | 0.83 | 0.28 | 0.85 | 0.36 |
| | | | 3 | individual median | 11 | 0.81 | 0.28 | 0.81 | 0.36 |
| | | | 4 | individual median | 13 | 0.83 | 0.34 | 0.84 | 0.42 |
| | | | 5 | individual median | 13 | 0.82 | 0.34 | 0.84 | 0.42 |
| | $JC_{NC}$ | max | 1 | individual median | 8 | 0.87 | 0.33 | 0.88 | 0.45 |
| | | | 2 | local median | 10 | 0.87 | 0.36 | 0.88 | 0.44 |
| | | | 3 | local mean | 11 | 0.86 | 0.34 | 0.87 | 0.48 |
| | | | 4 | local median | 3 | 0.87 | 0.25 | 0.88 | 0.32 |
| | | | 5 | local median | 5 | 0.87 | 0.27 | 0.88 | 0.30 |
| | | mean | 1 | individual median | 9 | 0.88 | 0.39 | 0.87 | 0.47 |
| | | | 2 | local median | 10 | 0.87 | 0.36 | 0.89 | 0.45 |
| | | | 3 | local sum | 8 | 0.87 | 0.31 | 0.88 | 0.41 |
| | | | 4 | local median | 3 | 0.87 | 0.26 | 0.88 | 0.36 |
| | | | 5 | local sum | 4 | 0.88 | 0.27 | 0.90 | 0.36 |
| NR | $JC_{CC}$ | – | 1 | global median | 4 | 0.71 | 0.18 | 0.70 | 0.26 |
| | | | 2 | individual sum | 5 | 0.76 | 0.21 | 0.69 | 0.23 |
| | | | 3 | default | 3 | 0.70 | 0.19 | 0.80 | 0.38 |
| | | | 4 | global mean | 8 | 0.74 | 0.25 | 0.64 | 0.17 |
| | | | 5 | global median | 2 | 0.75 | 0.21 | 0.76 | 0.18 |
| | $JC_{NC}$ | max | 1 | global median | 1 | 0.74 | 0.17 | 0.73 | 0.19 |
| | | | 2 | default | 1 | 0.79 | 0.21 | 0.79 | 0.17 |
| | | | 3 | local mean | 2 | 0.73 | 0.18 | 0.87 | 0.42 |
| | | | 4 | global median | 2 | 0.79 | 0.18 | 0.76 | 0.19 |
| | | | 5 | global mean | 2 | 0.77 | 0.20 | 0.77 | 0.23 |
| | | mean | 1 | local mean | 2 | 0.72 | 0.15 | 0.73 | 0.16 |
| | | | 2 | local sum | 1 | 0.77 | 0.19 | 0.75 | 0.14 |
| | | | 3 | global median | 2 | 0.71 | 0.19 | 0.87 | 0.46 |
| | | | 4 | global mean | 2 | 0.78 | 0.19 | 0.75 | 0.21 |
| | | | 5 | local sum | 2 | 0.76 | 0.21 | 0.76 | 0.16 |

Table 3: Spectral partitioning parameter optimization via internal cross-validation on benchmark data sets (Kinase)

| Data set | Measure | Aggregation function | Fold | Internal CV | | | | External CV | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Optimal parameters | | AUC | AUPR | AUC | AUPR |
| | | | | thresholding | m | | | | |
| Kinase | $JC_{CC}$ | – | 1 | local median | 10 | 0.76 | 0.17 | 0.76 | 0.22 |
| | | | 2 | local median | 11 | 0.77 | 0.19 | 0.76 | 0.22 |
| | | | 3 | local sum | 10 | 0.75 | 0.17 | 0.80 | 0.26 |
| | | | 4 | individual sum | 9 | 0.77 | 0.17 | 0.80 | 0.25 |
| | | | 5 | local mean | 9 | 0.76 | 0.16 | 0.74 | 0.21 |
| | $JC_{NC}$ | max | 1 | global median | 4 | 0.84 | 0.21 | 0.84 | 0.27 |
| | | | 2 | individual mean | 5 | 0.84 | 0.25 | 0.85 | 0.30 |
| | | | 3 | global median | 5 | 0.84 | 0.24 | 0.85 | 0.36 |
| | | | 4 | default | 4 | 0.83 | 0.22 | 0.85 | 0.33 |
| | | | 5 | individual mean | 5 | 0.84 | 0.23 | 0.84 | 0.32 |
| | | mean | 1 | default | 3 | 0.85 | 0.25 | 0.85 | 0.32 |
| | | | 2 | local sum | 5 | 0.86 | 0.29 | 0.86 | 0.33 |
| | | | 3 | default | 5 | 0.85 | 0.27 | 0.87 | 0.37 |
| | | | 4 | individual mean | 5 | 0.85 | 0.26 | 0.86 | 0.37 |
| | | | 5 | individual sum | 5 | 0.85 | 0.25 | 0.85 | 0.35 |

Table 4: Louvain algorithm parameter optimization via internal cross-validation on benchmark data sets (Enzyme)

| Data set | Measure | Fold | Internal CV | | | External CV | |
|---|---|---|---|---|---|---|---|
| | | | optimal resolution | AUC | AUPR | AUC | AUPR |
| Enzyme | $JC_{CC}$ | 1 | 0.4 | 0.94 | 0.52 | 0.96 | 0.71 |
| | | 2 | 0.4 | 0.94 | 0.52 | 0.95 | 0.71 |
| | | 3 | 0.4 | 0.94 | 0.52 | 0.95 | 0.67 |
| | | 4 | 0.4 | 0.94 | 0.51 | 0.96 | 0.74 |
| | | 5 | 0.4 | 0.94 | 0.52 | 0.95 | 0.63 |
| | $JC_{NC}$ | 1 | 0.6 | 0.95 | 0.55 | 0.96 | 0.72 |
| | | 2 | 0.5 | 0.96 | 0.58 | 0.96 | 0.70 |
| | | 3 | 0.6 | 0.95 | 0.56 | 0.95 | 0.68 |
| | | 4 | 0.6 | 0.96 | 0.57 | 0.96 | 0.73 |
| | | 5 | 0.6 | 0.95 | 0.54 | 0.97 | 0.72 |

Table 5: Louvain algorithm parameter optimization via internal cross-validation on benchmark data sets (GPCR, IC, NR, Kinase)

| Data set | Measure | Fold | Internal CV | | | External CV | |
|---|---|---|---|---|---|---|---|
| | | | optimal resolution | AUC | AUPR | AUC | AUPR |
| GPCR | $JC_{CC}$ | 1 | 0.7 | 0.72 | 0.16 | 0.81 | 0.32 |
| | | 2 | 0.7 | 0.73 | 0.18 | 0.76 | 0.24 |
| | | 3 | 0.8 | 0.71 | 0.15 | 0.83 | 0.27 |
| | | 4 | 0.7 | 0.74 | 0.21 | 0.78 | 0.36 |
| | | 5 | 0.7 | 0.74 | 0.19 | 0.84 | 0.34 |
| | $JC_{NC}$ | 1 | 0.8 | 0.84 | 0.26 | 0.90 | 0.50 |
| | | 2 | 0.8 | 0.84 | 0.30 | 0.89 | 0.49 |
| | | 3 | 0.8 | 0.83 | 0.24 | 0.88 | 0.43 |
| | | 4 | 0.8 | 0.86 | 0.30 | 0.89 | 0.49 |
| | | 5 | 0.8 | 0.85 | 0.28 | 0.88 | 0.54 |
| IC | $JC_{CC}$ | 1 | 0.6 | 0.94 | 0.49 | 0.95 | 0.60 |
| | | 2 | 0.5 | 0.94 | 0.50 | 0.95 | 0.69 |
| | | 3 | 0.5 | 0.94 | 0.51 | 0.96 | 0.67 |
| | | 4 | 0.5 | 0.93 | 0.50 | 0.94 | 0.68 |
| | | 5 | 0.5 | 0.94 | 0.51 | 0.95 | 0.61 |
| | $JC_{NC}$ | 1 | 0.8 | 0.96 | 0.59 | 0.98 | 0.79 |
| | | 2 | 0.9 | 0.96 | 0.59 | 0.97 | 0.74 |
| | | 3 | 0.7 | 0.96 | 0.64 | 0.97 | 0.80 |
| | | 4 | 0.8 | 0.96 | 0.60 | 0.96 | 0.80 |
| | | 5 | 0.8 | 0.96 | 0.61 | 0.96 | 0.76 |
| NR | $JC_{CC}$ | 1 | 0.8 | 0.83 | 0.28 | 0.84 | 0.49 |
| | | 2 | 0.7 | 0.81 | 0.32 | 0.82 | 0.41 |
| | | 3 | 0.8 | 0.75 | 0.18 | 0.81 | 0.26 |
| | | 4 | 0.7 | 0.79 | 0.28 | 0.90 | 0.60 |
| | | 5 | 0.7 | 0.79 | 0.31 | 0.73 | 0.51 |
| | $JC_{NC}$ | 1 | 0.8 | 0.84 | 0.28 | 0.84 | 0.40 |
| | | 2 | 0.7 | 0.82 | 0.36 | 0.88 | 0.57 |
| | | 3 | 0.8 | 0.77 | 0.24 | 0.73 | 0.31 |
| | | 4 | 0.7 | 0.79 | 0.32 | 0.89 | 0.57 |
| | | 5 | 0.8 | 0.83 | 0.28 | 0.70 | 0.26 |
| Kinase | $JC_{CC}$ | 1 | 0.7 | 0.72 | 0.14 | 0.73 | 0.19 |
| | | 2 | 0.7 | 0.72 | 0.14 | 0.74 | 0.20 |
| | | 3 | 0.7 | 0.73 | 0.15 | 0.74 | 0.20 |
| | | 4 | 0.6 | 0.73 | 0.15 | 0.71 | 0.17 |
| | | 5 | 0.7 | 0.73 | 0.15 | 0.72 | 0.18 |
| | $JC_{NC}$ | 1 | 0.9 | 0.90 | 0.38 | 0.91 | 0.54 |
| | | 2 | 0.9 | 0.90 | 0.37 | 0.92 | 0.50 |
| | | 3 | 0.9 | 0.90 | 0.39 | 0.91 | 0.49 |
| | | 4 | 0.9 | 0.90 | 0.38 | 0.91 | 0.50 |
| | | 5 | 0.9 | 0.89 | 0.40 | 0.91 | 0.47 |

Table 6: Spectral partitioning parameter optimization via internal cross-validation on generic data sets

| Data set | Measure | Fold | Internal CV | | | | External CV | |
|---|---|---|---|---|---|---|---|---|
| | | | Optimal parameters | | AUC | AUPR | AUC | AUPR |
| | | | thresholding | m | | | | |
| MovieLens | $JC_{CC}$ | 1 | local median | 12 | 0.82 | 0.23 | 0.80 | 0.28 |
| | | 2 | individual median | 11 | 0.82 | 0.18 | 0.83 | 0.22 |
| | | 3 | local median | 10 | 0.82 | 0.23 | 0.82 | 0.25 |
| | | 4 | local median | 11 | 0.82 | 0.22 | 0.83 | 0.27 |
| | | 5 | local median | 13 | 0.82 | 0.24 | 0.83 | 0.26 |
| | $JC_{NC}$ | 1 | local median | 6 | 0.88 | 0.29 | 0.89 | 0.39 |
| | | 2 | local median | 5 | 0.88 | 0.29 | 0.88 | 0.37 |
| | | 3 | local median | 5 | 0.88 | 0.30 | 0.89 | 0.36 |
| | | 4 | individual median | 4 | 0.88 | 0.30 | 0.89 | 0.38 |
| | | 5 | local median | 6 | 0.88 | 0.30 | 0.88 | 0.36 |
| Unicodelang | $JC_{CC}$ | 1 | individual median | 54 | 0.70 | 0.05 | 0.72 | 0.02 |
| | | 2 | individual median | 54 | 0.68 | 0.05 | 0.65 | 0.01 |
| | | 3 | individual median | 55 | 0.69 | 0.03 | 0.68 | 0.02 |
| | | 4 | individual median | 54 | 0.70 | 0.04 | 0.66 | 0.02 |
| | | 5 | individual median | 57 | 0.68 | 0.02 | 0.68 | 0.01 |
| | $JC_{NC}$ | 1 | individual mean | 3 | 0.79 | 0.14 | 0.79 | 0.13 |
| | | 2 | local median | 9 | 0.77 | 0.13 | 0.77 | 0.11 |
| | | 3 | individual sum | 16 | 0.78 | 0.13 | 0.78 | 0.16 |
| | | 4 | default | 16 | 0.79 | 0.13 | 0.78 | 0.19 |
| | | 5 | global mean | 7 | 0.78 | 0.15 | 0.78 | 0.25 |

Table 7: Louvain algorithm parameter optimization via internal cross-validation on generic data sets

| Data set | Measure | Fold | Internal CV | | | External CV | |
|---|---|---|---|---|---|---|---|
| | | | optimal resolution | AUC | AUPR | AUC | AUPR |
| MovieLens | $JC_{CC}$ | 1 | 0.3 | 0.79 | 0.18 | 0.79 | 0.23 |
| | | 2 | 0.3 | 0.79 | 0.18 | 0.77 | 0.22 |
| | | 3 | 0.3 | 0.80 | 0.19 | 0.76 | 0.23 |
| | | 4 | 0.3 | 0.79 | 0.18 | 0.77 | 0.22 |
| | | 5 | 0.3 | 0.79 | 0.18 | 0.77 | 0.22 |
| | $JC_{NC}$ | 1 | 0.6 | 0.88 | 0.26 | 0.88 | 0.34 |
| | | 2 | 0.8 | 0.88 | 0.27 | 0.88 | 0.35 |
| | | 3 | 0.6 | 0.88 | 0.26 | 0.89 | 0.34 |
| | | 4 | 0.7 | 0.88 | 0.26 | 0.88 | 0.33 |
| | | 5 | 0.6 | 0.88 | 0.26 | 0.88 | 0.33 |
| Unicodelang | $JC_{CC}$ | 1 | 0.6 | 0.73 | 0.10 | 0.72 | 0.08 |
| | | 2 | 0.6 | 0.73 | 0.09 | 0.73 | 0.07 |
| | | 3 | 0.6 | 0.73 | 0.09 | 0.74 | 0.10 |
| | | 4 | 0.6 | 0.73 | 0.09 | 0.69 | 0.09 |
| | | 5 | 0.6 | 0.73 | 0.10 | 0.73 | 0.15 |
| | $JC_{NC}$ | 1 | 0.7 | 0.80 | 0.14 | 0.80 | 0.12 |
| | | 2 | 0.7 | 0.79 | 0.13 | 0.80 | 0.13 |
| | | 3 | 0.8 | 0.79 | 0.14 | 0.84 | 0.16 |
| | | 4 | 0.7 | 0.81 | 0.11 | 0.79 | 0.16 |
| | | 5 | 0.7 | 0.80 | 0.14 | 0.79 | 0.25 |

## APPENDIX 2

Table 8: Evaluation of the performance of the majority vote of different classifiers for different feature sets and different ratios of FHs and iFHs in the test data

| Setting | Algorithm | Features | Accuracy | | AUC score | | Precision | | Recall | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ |
| Balanced | Random Forests | subgraphs | 37.72 | 2.49 | 0.38 | 0.02 | 0.43 | 0.02 | 0.75 | 0.05 |
| | | numeric | 81.67 | 2.92 | 0.82 | 0.03 | 0.86 | 0.03 | 0.76 | 0.05 |
| | | subgraphs+numeric | 83.94 | 15.93 | 0.84 | 0.16 | 0.90 | 0.16 | 0.81 | 0.03 |
| | SVM | subgraphs | 78.5 | 16.8 | 0.79 | 0.17 | 0.94 | 0.19 | 0.67 | 0.05 |
| | | numeric | 51.56 | 0.78 | 0.52 | 0.01 | 0.87 | 0.32 | 0.03 | 0.01 |
| | | subgraphs+numeric | 52.22 | 1.26 | 0.52 | 0.01 | 0.87 | 0.32 | 0.05 | 0.02 |
| | Naive Bayes | subgraphs | 61.83 | 16.59 | 0.62 | 0.17 | 0.92 | 0.24 | 0.34 | 0.08 |
| | | numeric | 68.83 | 2.32 | 0.69 | 0.02 | 0.67 | 0.03 | 0.76 | 0.03 |
| | | subgraphs+numeric | 68.67 | 16.32 | 0.69 | 0.16 | 0.93 | 0.21 | 0.47 | 0.09 |
| | Logistic Regression | subgraphs | 42.06 | 16.18 | 0.42 | 0.16 | 0.48 | 0.18 | 0.74 | 0.04 |
| | | numeric | 82.83 | 3.84 | 0.83 | 0.04 | 0.83 | 0.05 | 0.83 | 0.04 |
| | | subgraphs+numeric | 77.78 | 12.73 | 0.78 | 0.13 | 0.77 | 0.12 | 0.85 | 0.03 |
| | Stochastic Gradient Descent | subgraphs | 38.56 | 2.69 | 0.39 | 0.03 | 0.43 | 0.02 | 0.77 | 0.05 |
| | | numeric | 71.22 | 2.59 | 0.71 | 0.03 | 0.72 | 0.05 | 0.72 | 0.07 |
| | | subgraphs+numeric | 71.39 | 3.4 | 0.71 | 0.03 | 0.71 | 0.04 | 0.73 | 0.04 |
| Slight imbalance | Random Forests | subgraphs | 6.93 | 0.31 | 0.38 | 0.02 | 0.07 | 0.00 | 0.76 | 0.03 |
| | | numeric | 85.79 | 1.16 | 0.81 | 0.02 | 0.37 | 0.02 | 0.76 | 0.03 |
| | | subgraphs+numeric | 85.86 | 27.55 | 0.84 | 0.15 | 0.61 | 0.19 | 0.81 | 0.02 |
| | SVM | subgraphs | 87.85 | 28.56 | 0.78 | 0.15 | 0.91 | 0.29 | 0.66 | 0.04 |
| | | numeric | 91.18 | 0.21 | 0.52 | 0.01 | 0.87 | 0.19 | 0.03 | 0.02 |
| | | subgraphs+numeric | 91.29 | 0.28 | 0.52 | 0.01 | 0.89 | 0.17 | 0.05 | 0.03 |
| | Naive Bayes | subgraphs | 84.8 | 28.13 | 0.61 | 0.13 | 0.91 | 0.30 | 0.33 | 0.09 |
| | | numeric | 63.29 | 1.45 | 0.69 | 0.02 | 0.17 | 0.01 | 0.76 | 0.04 |
| | | subgraphs+numeric | 85.8 | 28.2 | 0.68 | 0.13 | 0.87 | 0.29 | 0.45 | 0.08 |
| | Logistic Regression | subgraphs | 6.67 | 0.28 | 0.37 | 0.02 | 0.07 | 0.00 | 0.73 | 0.03 |
| | | numeric | 81.62 | 0.79 | 0.82 | 0.02 | 0.31 | 0.01 | 0.83 | 0.04 |
| | | subgraphs+numeric | 71.81 | 22.46 | 0.78 | 0.12 | 0.26 | 0.07 | 0.85 | 0.03 |
| | Stochastic Gradient Descent | subgraphs | 6.89 | 0.36 | 0.38 | 0.02 | 0.07 | 0.00 | 0.76 | 0.04 |
| | | numeric | 71.66 | 4.76 | 0.71 | 0.02 | 0.20 | 0.02 | 0.69 | 0.05 |
| | | subgraphs+numeric | 71.8 | 4.38 | 0.71 | 0.02 | 0.20 | 0.02 | 0.69 | 0.07 |
| Severe imbalance | Random Forests | subgraphs | 0.74 | 0.04 | 0.37 | 0.02 | 0.01 | 0.00 | 0.75 | 0.04 |
| | | numeric | 87.11 | 0.58 | 0.82 | 0.03 | 0.06 | 0.00 | 0.77 | 0.05 |
| | | subgraphs+numeric | 86.71 | 30.18 | 0.84 | 0.15 | 0.17 | 0.06 | 0.81 | 0.03 |
| | SVM | subgraphs | 89.77 | 31.31 | 0.78 | 0.16 | 0.90 | 0.31 | 0.67 | 0.04 |
| | | numeric | 99 | 0.03 | 0.52 | 0.01 | 0.41 | 0.16 | 0.03 | 0.02 |
| | | subgraphs+numeric | 98.99 | 0.02 | 0.52 | 0.01 | 0.38 | 0.13 | 0.04 | 0.02 |
| | Naive Bayes | subgraphs | 89.45 | 31.33 | 0.62 | 0.17 | 0.90 | 0.32 | 0.34 | 0.09 |
| | | numeric | 62.73 | 0.67 | 0.69 | 0.04 | 0.02 | 0.00 | 0.76 | 0.07 |
| | | subgraphs+numeric | 89.39 | 31.28 | 0.67 | 0.17 | 0.65 | 0.23 | 0.44 | 0.07 |
| | Logistic Regression | subgraphs | 0.73 | 0.05 | 0.37 | 0.02 | 0.01 | 0.00 | 0.74 | 0.05 |
| | | numeric | 81.34 | 0.49 | 0.82 | 0.01 | 0.04 | 0.00 | 0.83 | 0.02 |
| | | subgraphs+numeric | 71.56 | 24.76 | 0.78 | 0.12 | 0.04 | 0.01 | 0.84 | 0.02 |
| | Stochastic Gradient Descent | subgraphs | 0.74 | 0.04 | 0.38 | 0.02 | 0.01 | 0.00 | 0.75 | 0.04 |
| | | numeric | 72.95 | 4.71 | 0.71 | 0.03 | 0.02 | 0.00 | 0.69 | 0.10 |
| | | subgraphs+numeric | 72.63 | 4.43 | 0.71 | 0.03 | 0.03 | 0.00 | 0.70 | 0.08 |

# Link prediction in bipartite multi-layer networks, with an application to drug-target interaction prediction

**Abstract:** Many aspects from real life with bi-relational structure can be modeled as bipartite networks. This modeling allows the use of some standard solutions for prediction and/or recommendation of new relations between these objects in such networks. Known as the link prediction task, it is a widely studied problem in network science for single graphs, networks assuming one type of interaction between vertices. For multi-layer networks, allowing more than one type of edges between vertices, the problem is not yet fully solved.

The motivation of this thesis comes from the importance of an application task, drug-target interaction prediction. Searching valid drug candidates for a given biological target is an essential part of modern drug development. In this thesis, the problem is modeled as link prediction in a bipartite multi-layer network. Modeling the problem in this setting helps to aggregate different sources of information into one single structure and as a result to improve the quality of link prediction.

The thesis mostly focuses on the problem of link prediction in bipartite multi-layer networks and makes two main contributions on this topic. The first contribution provides a solution for solving link prediction in the given setting without limiting the number and type of networks, the main constrains of the state of the art methods. Modeling random walk in the fashion of PageRank, the algorithm that we developed is able to predict new interactions in the network constructed from different sources of information. The second contribution, which solves link prediction using community information, is less straight-forward and more dependent on fixing the parameters, but provides better results. Adopting existing community measures for link prediction to the case of bipartite multi-layer networks and proposing alternative ways for exploiting communities, the method offers better performance and efficiency. Additional evaluation on the data of a different origin than drug-target interactions demonstrate the genericness of proposed approach.

In addition to the developed approaches, we propose a framework for validation of predicted interactions founded on an external resource. Based on a collection of biomedical concepts used as a knowledge source, the framework is able to perform validation of drug-target pairs using proposed confidence scores. An evaluation of predicted interactions performed on unseen data shows effectiveness of this framework.

At the end, a problem of identification and characterization of promiscuous compounds existing in the drug development process is discussed. The problem is solved as a machine learning classification task. The contribution includes graph mining and sampling approaches. In addition, a graphical interface was developed to provide feedback of the result for experts.

***Key words:*** *machine learning, data mining, network science, multi-layer networks*

# Prédiction de liens dans les réseaux bipartis multicouche, avec une application à la prédiction d'interaction médicament-cible thérapeutique

**Résumé:** De nombreux problèmes réels relèvent d'une structure bi-relationnelle et peuvent être modélisés suivant des réseaux bipartis. Une telle modélisation permet l'utilisation de solutions standards pour la prédiction et/ou la recommandation de nouvelles relations entre objets de ces réseaux. La tâche de prédiction de liens est un problème largement étudié dans les réseaux simples, c'est-à-dire les réseaux avec un seul type d'interaction entre sommets. Cependant, pour les réseaux multicouche (i.e. réseaux avec plusieurs types d'arêtes entre sommets), ce problème n'est pas encore entièrement résolu.

Cette thèse est motivée par l'importance d'une tâche réelle, à savoir la prédiction d'interaction entre un médicament et une cible thérapeutique. La recherche de candidats médicaments prometteurs pour une cible thérapeutique biologique donnée est une partie essentielle de la conception d'un médicament moderne. Dans cette thèse, nous modélisons ce problème comme une tâche de prédiction de lien dans un réseau multicouche biparti. Cette modélisation du problème permet de rassembler différentes sources d'information en une seule structure et ainsi d'améliorer la qualité de la prédiction d'un lien.

Cette thèse se concentre sur le problème de la prédiction de liens dans les réseaux multicouches bipartis et apporte deux contributions principales à ce sujet. La première contribution est une solution pour résoudre la prédiction de liens sans limiter le nombre et le type de réseaux, ce qui est le principal défaut des méthodes de l'état de l'art. L'algorithme que nous avons développé modélise une marche aléatoire à la manière du PageRank et est capable de prédire de nouvelles interactions dans le réseau que nous construisons à partir de différentes sources d'information. La deuxième contribution, qui porte aussi sur ce problème, s'appuie sur les méthodes de détection de communautés. Cette solution, moins immédiate et plus dépendante du choix des valeurs des paramètres, donne de meilleurs résultats. Pour cela, nous adaptons des mesures utilisées pour la détection de communautés à la problématique de la prédiction de liens dans les réseaux multicouche bipartis et nous développons de nouvelles méthodes associant des communautés pour la prédiction de liens. Nous évaluons aussi nos méthodes sur des données autres que celles des interactions entre médicaments et cibles thérapeutiques montrant ainsi le caractère générique de notre approche.

D'autre part, nous proposons un protocole expérimental de validation des interactions prédites reposant sur l'exploitation de ressources externes. Fondé sur une collection de concepts biomédicaux utilisés comme source de connaissances, ce protocole effectue une validation des paires de médicaments-cibles thérapeutiques qui sont prédites à partir de scores de confiance que nous avons définis. Une évaluation des interactions prédites sur des données tests montre l'efficacité de ce protocole.

Enfin, nous nous intéressons au problème de l'identification et de la caractérisation de composés promiscues qui existe dans le processus de développement de médicaments. Nous modélisons ce problème comme une tâche de classification et le résolvons par l'apprentissage automatique. Notre contribution repose sur une approche d'exploration de graphes et d'échantillonnage. De plus, nous avons développé une interface graphique pour fournir un retour d'information aux experts sur les résultats.

***Mots-clés:*** *apprentissage automatique, exploration de données, théorie des réseaux, réseaux multicouches*