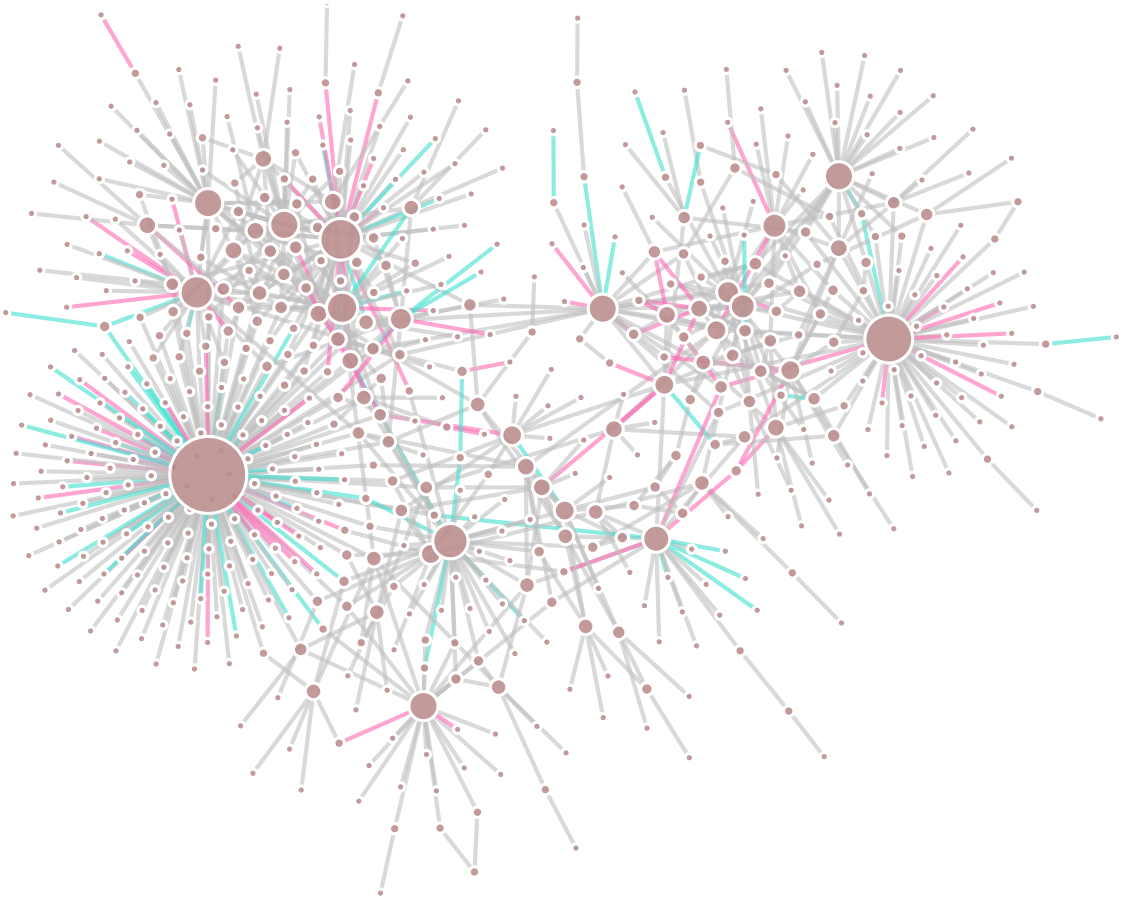
CHALMERS
UNIVERSITY OF TECHNOLOGY



Discovering Novel Chemical Reactions

Using Graph Neural Networks for Link Prediction in a Knowledge Graph

Master's Thesis in Complex Adaptive Systems

Emma Rydholm and Emma Svensson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's Thesis 2021

Discovering Novel Chemical Reactions

Using Graph Neural Networks for Link Prediction in a Knowledge Graph

Emma Rydholm and Emma Svensson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
Gothenburg, Sweden 2021

Discovering Novel Chemical Reactions
Using Graph Neural Networks for Link Prediction
in a Knowledge Graph
EMMA RYDHOLM & EMMA SVENSSON

Supervisor: Tomas Bastys, AstraZeneca
Thierry Kogej, AstraZeneca
Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Examiner: Peter Damaschke, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of a sub-component from the knowledge graph of molecular template vertices and reacts edges. Sizes of vertices determined by total vertex degree and edges colored by training set (grey), validation set (pink), and test set (turquoise).

Typeset in LaTeX
Gothenburg, Sweden 2021

Discovering Novel Chemical Reactions
Using Graph Neural Networks for Link Prediction in a Knowledge Graph
EMMA RYDHOLM & EMMA SVENSSON
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Accurately predicting chemical reactions can facilitate the search for optimal synthesis routes in a chemical reaction network and as a consequence expedite the lengthy drug discovery process. As an effort in this direction, this work aims to explore AstraZeneca's chemical knowledge graph by two complementary analyses. In a first part, graph theory related statistics is employed as a means to gain insights about the chemical reaction graph at AstraZeneca. Significant differences are observed between this internal reaction graph and the one based on the public dataset of United States patents as well as other reaction graphs discussed in literature. Secondly, a link prediction model is applied to and evaluated on AstraZeneca's chemical reaction graph, in order to suggest new potential chemical reactions. In order to successfully accomplish this task, an existing link prediction model is adapted and trained. The test results are then compared to heuristic baselines, showing that the proposed implementation substantially exceeds what can be achieved with heuristic methods. One of the contribution from this research is a comparison between different ways to sample the ground truth class of non-existing links for training and evaluation. The choice of method for this task is shown to have an impact on the final predictions. Finally, a set of promising, predicted reactions are suggested and is currently under further investigation at AstraZeneca.

Keywords: Link Prediction, Graph Neural Networks, Knowledge Graph, Chemical Reaction Graph, Graph Analysis, Synthesis Prediction, Drug Discovery.

Acknowledgements

This project would not have been possible without our supervisors Tomas Bastys and Thierry Kogej, whose engagement, ideas and assistance has been incredible valuable to us. We thank Tomas especially since his own research has enabled this thesis and Thierry for helping us connect our work to the drug discovery context. Many thanks also goes to Christos Kannas who has saved us from hours of frustration and whose work has contributed to this project. Special thanks to Per-Ola Norrby whose chemical expertise have been crucial for evaluating the results. We have valued the time spent at AstraZeneca and would also like to direct our gratitude to Ola Engkvist and Esben Jannik Bjerrum for their insights and guidance as well as the rest of the team at AstraZeneca, who we have had the fortuity to meet during the last six months. Finally, we thank Morteza Haghir Chehreghani, our supervisor at Chalmers, and Peter Damaschke, our examiner, who have provided us with helpful directions along the course of this project.

Emma Rydholm and Emma Svensson, Gothenburg, May 2021

Contents

List of Figures

List of Tables

Acronyms

AA Adamic-Adar
AP Average Precision
AUC Area Under the ROC-curve

BCE Binary Cross-Entropy

CC Connected Component
CCDF Complement Cumulative Distribution Function
CN Common Neighbors
CNN Convolutional Neural Network

DGCNN Deep Graph Convolutional Neural Network
DRNL Double Radius Node Labelling

ELN Electronic Laboratory Notebook

FN False Negative
FP False Positive

GAT Graph Attention Network
GNN Graph Neural Network

JI Jaccard Index

MAP Mean Average Precision
MW Molecular Weight

OGB Open Graph Benchmark

ReLU Rectified Linear Unit
ROC Receiver Operating Characteristic

SCC Strongly Connected Component
SEAL learning from Subgraphs, Embeddings and Attributes for Link prediction
SMARTS SMiles ARbitrary Target Specification
SMILES Simplified Molecular Input Line Entry System

TN True Negative
TP True Positive

USPTO United States Patent and Trademark Office

VGAE Variational Graph Auto-Encoders

WL Weisfeiler-Lehman

1

Introduction

Being able to accurately predict the chemical accessibility of potential bioactive compounds has the potential to transform the way and the speed at which we are progressing in drug discovery programs. The emergence of large scale databases of chemical data during the last decades, has enabled the possibility to increase the efficiency and speed of drug discovery by applying machine learning methods at various synthesis-related prediction tasks. Specifically, synthesis prediction can help the chemist to find optimal pathways, called *synthesis routes*, from the available molecules to drug targets [1]. This thesis is inscribed at the Molecular AI department at AstraZeneca, with the purpose to impact the company's drug discovery process by generating new insights from chemical reaction data.

A thorough review of the previous machine learning and artificial intelligence methods that have been used in synthesis related prediction tasks, has been published by the team at AstraZeneca [2]. These methods were found to focus on either predicting possible reaction outcomes, e.g. [3], or optimal synthesis routes, e.g. [4], but not much research was found related to also predicting new chemical reactions and/or de novo products. The review states that so far, the relative performance of the methods is difficult to assess because of the limited quality of the public data.

Chemical reactions are suitably represented as a graph because of their linkage between reacting molecules, e.g reactants, and the resulting reaction entities, e.g products. Previous studies have shown that large scale chemical reaction graphs typically have some specific properties, such as exhibiting small-world behaviour which in short means that any molecule can on average be reached from any other with relatively few reactions [5, 6]. As part of this project, an analysis of an internal reaction graph is carried out together with a comparison of the findings with previously published work, to better understand the chemistry performed at AstraZeneca.

A graph representation of a chemical reaction space, enables chemical reactions to be encoded as links in the graph and consequently, link prediction methods can be used to predict novel chemical reactions. Link prediction is, as the name suggests, the task of predicting a linkage between two vertices in a graph. Segler & Waller described, in [7], their research where they applied heuristic link prediction in a knowledge graph for predicting chemical reactions. In this approach they also included a chemical reasoning algorithm that ensured chemically meaningful reactions. The results showed that their model outperformed solely rule-based systems

in predicting new chemical reaction.

As it is highly interesting to find more efficient pathways in a chemical reaction graph, we in this project successfully apply a link prediction method to AstraZeneca's chemical reaction graph, as a means to discover novel chemical reactions.

1.1 Scope

As such, this thesis aims to extend the state-of-the-art research regarding link prediction in chemistry-based knowledge graphs. Particularly, this is done through the development of a machine learning model for predicting novel chemical reactions in a subset of AstraZeneca's internal Electronic Laboratory Notebook (ELN) reaction knowledge graph [1]. The purpose of such predictions is to give valuable insights to the pharmaceutical company's manufacturing process. As an example the predictions can help to identify optimal synthesis routes to newly discovered chemical compounds.

Development of the link prediction model for this scope is done through a number of phases, all with their respective challenges and advantages. A graph structure is chosen such that the task can be performed effectively and resulting outcomes can be accessed and interpreted. It is also important to understand the data, and thus an analysis is conducted regarding statistical properties of the graph. This includes a comparison with previous published work on other reaction graphs and a similar analysis of the reaction graph where the chemical reactions are derived from United States patents, i.e. United States Patent and Trademark Office (USPTO) [8]. The second aspect of the problem is to take the currently most suitable method for link prediction and transfer it to the given context. Attempts are made to improve the method through alternations in training data as well as modifications to the original model architecture. Finally, predictions are evaluated in terms of traditional metrics for binary classification and to some extent in terms of the reactions chemical feasibility. As a part of the evaluation, the top predicted reaction are also interpreted by an experienced chemist within the team at AstraZeneca.

1.1.1 Research Questions

Concisely, the problem formulation investigated in this project can be summarized into the following two points.

1. *What interesting and relevant statistical properties can be observed in AstraZeneca's internal knowledge graph of chemical reactions? How does these differ from the properties of similar, public reaction graphs?*

2. *Can link prediction using a graph neural networks-based model be used to find meaningful, novel synthesis routes between chemical compounds? How should the training data be constructed and which molecule attributes yield the best predictions?*

1.1.2 Limitation

As part of the phase to choose a graph structure suitable for the link prediction task, a condensed version of the reaction graph was settled on by AstraZeneca researchers. In this graph version, the molecules and reactions are encoded in a more general way using so-called *templates*, described in Section 2.1. This has various advantages for the project, such as keeping the network to a manageable size as well as having the potential to lead to more novelty due to the generalization. As a result, a predicted reaction is not restricted to a specific molecule but a representation which can fit multiple molecules. This generalization can in someways also be a drawback as additional tracking in the complete graph is required to find molecules that could be potential reactants in the predicted reaction. In some cases there could even be a large set of such molecules. These steps, as well as additional in depth evaluation of the chemical feasibility of the predicted reactions, are out of scope for this project. Using this template-based approach further restricts the link prediction model to binary reactions.

The reaction data within AstraZeneca's ELN contains a large variety of chemical information about reactions and the chemical compounds involved. This includes for example the reaction type, the chemical context within which the reaction took place such as the temperature, and the set of reagents used such as catalyst and solvent. Considering all of this information would be overwhelming and instead the scope was limited to the information contained in so-called fingerprint vectors of molecules. Nevertheless, the model is flexible in the attributes attached to vertices and thus future experiments could be made with other types of information included, as deemed of interest. Further, the current model setup has been trained on a graph where all vertices are included from start and has not been tested for prediction of links to newly introduced vertices. The current model is therefore limited to predict links between the vertices currently present in the network.

In the interest of time, the optimization of the developed link prediction model could only be performed on one of the scenarios of graph structure and training data considered. The result was then directly transferred to other scenarios. Also, due to the vast set of options in the model (e.g. *hyperparameters*) an assumption is made regarding independence. This enable us to focus the analysis on the individual effects of the hyperparameters on the training performance, whereas joint effects are ignored.

1.2 Thesis Outline

Overall the report contains two main parts, the study of the graph and the link prediction including its implementation, optimization and evaluation. To fully comprehend the different aspects of this project, the report includes three theory chapters. The first covers all cheminformatic concepts used in this study. Secondly, some graph theoretical concepts are introduced, specifically ones related to the graph environment and graph statistics. Lastly, relevant topics for the link prediction task are presented. A description of the methods used in this work follows in the suc-

ceeding chapter, which is divided into three sections. First the graph architecture is described, followed by a description of how the graph analysis is performed. Finally, the implementation of the link prediction model is outlined. The results of the respective parts, are divided into two chapters where the analysis results and link prediction results are described and discussed separately. The two final chapters includes the conclusions from this work and a proposal for future works.

1.3 Contributions

Ultimately, this thesis contributes with a complete, end-to-end algorithm and trained model for prediction of novel chemical reactions in AstraZeneca's internal knowledge graph. This has the potential to impact the efforts required by chemists, with regards to exploring reactions since the model can give initial hints as to which new reactions to look at.

Regarding the method used for the link prediction task, we contribute with conclusions on how the choice of training data affects the outcome performance of the model in a context where missing links can have different underlying meanings. This was done through a number of experiments exploring different ways to setup the so-called *negative class* in the data, namely the missing links. A few variations of the model were also implemented and evaluated, which can be of interest to future research.

Additionally, an analysis about the graph, built on the internal chemistry at AstraZeneca, has been performed. This provided insights on the type of chemicals and chemistry used by the company and how this compares to a representative and comprehensive public patent reaction database, USPTO.

2

Cheminformatics

Cheminformatics is a discipline which deals with the description and the utilization of chemistry data using computer based methods. The reaction graphs used in this project consists of molecules and reactions vertices, which are encoded in a way so that it is possible to identify each unique molecule and reaction. In this section, the cheminformatic concepts used in this study will be presented. More specifically, Section 2.1 introduces how molecules and reactions can be represented for computer-based tasks and Section 2.2 describes some molecular descriptors.

2.1 Chemical Representation

As part of this thesis a number of chemical representations was needed for explaining molecule's involvement in reactions and thus their part in synthesis routes. A common way to represent a molecule is using a *molecular graph*, see example of the common drug paracetamol's such representation in Figure 2.1.

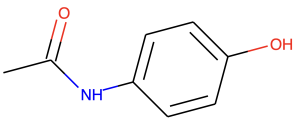


Figure 2.1: Molecular graph of paracetamol. The atoms are represented by their atomic symbol except carbon atoms, witch are implied by line intersection thus not explicitly written. Also, the hydrogen atoms which are bonded to the carbons are implicit.

Although, the humanly readable molecular graph is useful in many applications it is not optimal when molecules need to be processed by computer algorithms. In particular, the choice of representation is crucial for feeding the data to the neural network, where a numerical representation is often preferred. A large range of molecular representations exist for this purpose, here we only present the ones used in this project.

String representation A widely used chemical language is SMILES, for Simplified Molecular Input Line Entry System. It is derived from the molecule model

based on a set of rules and uses ASCII characters to represent atoms and bonds. Atoms are represented by their atomic symbol and when they belong to a so-called *aromatic systems* they are most commonly written using lower-case letters. Normally, implicit hydrogen atoms are omitted since they are given by the other atoms and branches are enclosed inside parenthesis.

SMILES denotes a molecular structure and hence is not necessarily a distinct representation for a given molecule. There are however tools to create unambiguity in the form of canonicalization [9]. The SMILES representation of the example molecule from Figure 2.1 is

```
CC(=O)Nc1ccc(O)cc1
```

An extension of this representation is the so called SMARTS, abbreviation for SMiles ARbitrary Target Specification. All SMILES are valid SMARTS, but the opposite is not true. The main difference from SMILES, is that this representation can be used as a query language to specify and search for substructures within a molecule, using additional rules to those used for SMILES. Another difference to SMILES is that SMARTS allows for more flexibility, by including logical operators that can symbolize one out of two atoms [10]. While both these representations can be used to describe exact chemical reactions, a third representation called SMIRKS has been developed for the purpose of describing generic transformations occurring during chemical reactions [10]. This language follows rules derived from both SMILES and SMARTS and includes some additional requirements for reactants and products sides.

SMILES, SMARTS and SMIRKS have also been extended to handle reactions. The example reaction in Figure 2.2 can be represented in SMILES language as

```
CC(=O)OC(=O)C.Nc1ccc(O)cc1>>CC(=O)Nc1ccc(O)cc1
```

where the dot is used to symbolize the disconnected structures and the arrow implies the reaction direction. The green part of the reaction SMILES represents the SMILES for acetic anhydride, the purple represents p-aminophenol and the black part is paracetamol.

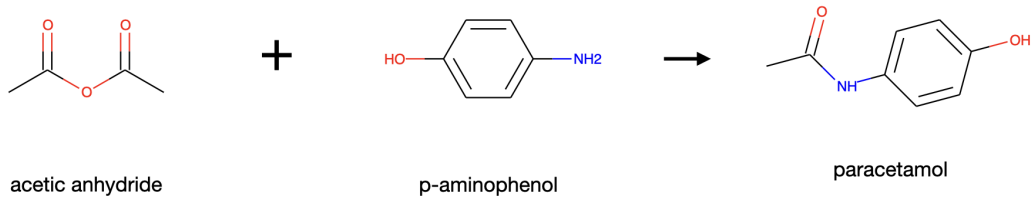


Figure 2.2: Example reaction showing the last step in a synthesis route to paracetamol. Note that this is an unbalanced reaction.

Reaction template Chemical reactions can be described in a more general way by reaction templates. These use the SMIRKS language and includes the reaction center, called *reaction site*, and some variable radius of its neighborhood [11]. Because of the generalization, the same reaction template can describe multiple reactions involving similar reactants and products, since the region which are not included in the template can be varied. The matching of atoms from one side of the reaction to the other relies on a process called *atom mapping* carried out by dedicated algorithms. These templates can also include functional and protective groups which can have an effect on reaction taking place or not [12]. With a radius of two, the example reaction from Figure 2.2 has the following reaction template,

```
C-C(=O)-O-[C;H0;D3;+0:1](-[C;D1;H3:2])=[O;D1;H0:3].[NH2;D1;+0:4]-[c:5]
    >>[C;D1;H3:2]-[C;H0;D3;+0:1](=[O;D1;H0:3])-[NH;D2;+0:4]-[c:5]
```

A reaction template, can further be divided into *half-templates*, which describes individual molecules involved in the reaction and their transformation to the product. This half-template contains information about how the individual atoms and bonds are transformed to the product [13]. For the running example, the half templates are the following,

```
{C-C(=O)-O-[C;H0;D3;+0:1](-[C;D1;H3:2])=[O;D1;H0:3]>>
 [C;D1;H3:2]-[C;H0;D3;+0:1](=[O;D1;H0:3])-X

 [NH2;D1;+0:4]-[c:5] >> X-[NH;D2;+0:4]-[c:5]
```

where `X` symbolizes the missing part in each half-template. Two half-templates can further be combined into a reaction template. Consequently, if the half-templates cannot be combined this suggests that there cannot be a reaction between the reactants, in the given framework.

Molecular Fingerprint Molecules can also be represented by *molecular fingerprints* which are bit vectors of various fixed length, typically ranging from 500-4000. These were first developed for substructure searches and have more recently been used for similarity measurement, clustering and classification. A fingerprint includes information about the 2D structure of the molecule and what atoms and substructures are included. In particular, a 1 indicates presence of a specific substructure while a 0 indicates the absence of that substructure. There are several algorithms for constructing a fingerprint and a common way is the so-called *circular fingerprints* which are based on the Morgan algorithm. In short, the Morgan algorithm looks for the presence or absence of substructures in the circular neighborhood of one atom at a time. The depth of neighbors considered, is provided by the radius parameter [14].

2.2 Molecular Descriptors

Given a molecular identifier, it is straightforward to extract more information about the molecule, for example by using the cheminformatic toolkit RDKit. Molecules have many properties that impact their function and interaction with other molecules, some of these are important to acknowledge in the drug discovery process.

The complexity of a molecule can impact the part it plays in the drug synthesis route. Typically simple molecules are building-blocks and starting points of paths, whereas more complex molecules are found further into the synthesis routes. The Molecular Weight (MW) is the sum of all atomic masses of the atoms present in a molecule and is strongly correlated to the number of heavy atoms (i.e. all atoms apart hydrogen) it contains. Another descriptor that measure the molecular complexity, is the number of rings. As a chemical ring corresponds to a molecular entity made of several atoms arranges in a closed cycle structure.

In drug discovery it is important to be aware of the *molecular chirality*. A *chiral molecule* is a molecule that is not identical to its mirror image, these molecules are called a pair of *enantiomers*. The pair has the same composition of atoms and chemical properties but since they are not superimposable on each other, they are two distinct compounds. Chirality becomes important when chiral molecules react with other chiral compounds since the pharmacological function can be very different for the two enantiomers. Consequently, only one of the enantiomers might be suitable as a drug. A useful measure regarding the chirality of a molecule is the number of chiral centers included in the molecule, where a chiral center is an atom which has four different groups attached to it.

Another descriptor, which has been established as a relevant measure of molecular complexity in the literature [15], is the fraction of carbon atoms that are sp^3 hybridized, the Fsp3 value. Hybridization is explained in *Molecular Orbital Theory* and is a way to describe the distribution of electrons in a molecule by combining atomic orbitals. The hybridisation of a molecule is related to its geometrical shape. When the central atom is sp^3 hybridised the molecule is in a tetrahedral shape, meaning that four atoms are attached to the central atom with a bond angle of 109.5° [16].

For the target compounds of a drug synthesis process, it is also important to evaluate its drug-likeliness. This is typically done through Lipinski's Rule-of-Five [17], which gives a general idea of what values of certain physio-chemical properties are typical for compounds that are orally active drugs for humans. These rules include limits on the molecular weight, the number of hydrogen bond donors (HBD) and acceptors (HBA) and the logarithm of the n-octanol-water partition coefficient, the cLogP value. The polar surface area, PSA, of the molecule can also give an indication of the drug-likeliness [18].

3

Graph Theory

As a chemical reaction graph is a central part of this thesis, the following chapter gives the relevant background on graph theory. Section 3.1 formally defines all notations used. Section 3.2 introduces *knowledge graphs* in general and chemical reaction graphs in particular. Lastly, Section 3.3 is dedicated to scale-free graphs as well as some related graph metrics and properties.

3.1 Definitions

Here, some definitions and notations related to graphs are introduced.

- A graph $G(V, E)$ is defined as a set V of vertices and a set E of edges such that each edge connects two vertices in either a directed or undirected sense.
- Every edge $e_{i,j} \in E$ has a source vertex v_i and a target vertex v_j, where $v_i, v_j \in V$. The edge is written as $e_{i,j} = (v_i, v_j)$ or simply as $e_{i,j} = (i, j)$.
- In an *undirected graph* the edges lack a direction, therefore $e_{i,j} = (v_i, v_j) = (v_j, v_i) \quad \forall v_i, v_j \in V$, the same does not hold for a *directed graph*.
- Vertices connected by an edge are called *adjacent* or *neighbors*. A common representation of edges in a graph is using an *adjacency matrix*, where each element $a_{i,j}$ stores the boolean of whether or not an edge connects vertices i and j. The sparse representation of this matrix is called an *adjacency list*.
- The number of edges that connects a vertex is called its *degree* or *degree centrality* and the set of neighbors to vertex v_i can be written Γ_i.
- In a directed graph vertices can have three different degree properties, in-degree k_{in} and out-degree k_{out} being the sum of all incoming and outgoing edges respectively and total degree being the sum of the in-degree and out-degree [19].
- A graph is called *bipartite* if V is the union of two disjoint independent sets, see Figure 3.1. This is true if and only if it has no *odd cycles*.
- An odd cycle is a path, from one vertex back to itself using an odd number of edges.

- The density D of an undirected graph is calculated according to

$$D = \frac{2|E|}{|V|(|V|-1)}, \tag{3.1}$$

where $|E|$ and $|V|$ are the number of edges and vertices respectively. For a directed graph, D is divided by 2.

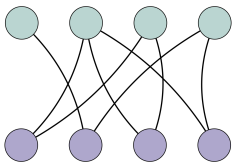


Figure 3.1: A bipartite graph. Green vertices only connect to purple vertices and vice versa.

When a graph describes a real-world problem, such as social relations or neurons in the human brain, the graph term is interchangeable with the term *network*. In this report the term *graph* will be used consistently to describe the chemical reaction graph, to avoid confusion with for example, neural networks. The terms vertex and *node* can be used interchangeably and the same holds for the terms edge and *link*, which can also be referred to as *connection* or *relation*.

3.2 Knowledge Graphs

Graphs can be extended to handle knowledge databases such that entries and their interlinkage can be stored in a free-formed semantic. This concept is called a *knowledge graph* in which vertices and edges can come in different classes and with various properties. In particular, edges come in triplets of information containing the source and target vertices and the relation between the two [20]. The information contained in each vertex is called its *attributes* and the flexibility of what can be included enables knowledge graphs for a wide variety of applications, including question answering and information retrieval [21].

Chemical reaction data is through its linkage between reactant and product compounds inherently suited to be exploited in a knowledge graph [11]. How to represent chemical reactions as a graph was explored by Bishop et al. [22], where the molecules are the vertices and the reactions are represented as directed edges that connect the vertices. An alternative way to encode chemical reactions in a knowledge graph, is to include reactions as their own vertices in addition to the molecule vertices, as can be seen in Figure 3.2. This alternative wiring scheme can be advantageous since it enables more information about the reactions to be encoded in the graph, for example each molecule's particular role can be described by edge labels [7].

Most of the concepts from general graph theory can be directly transferred when analyzing knowledge graphs. This includes various metrics, such as the average

shortest path length between all vertices in the graph and different measures of centrality. An important graph property, which also applies to knowledge graphs, is the so-called *scale-free* behaviour which describes how a graph scales with its size. Previous research has found a graph compromising of the largest dataset of organic chemistry reactions to be scale-free [6, 22]. Other features, commonly found in scale-free networks are a hierarchical structure and the presence of highly connected vertices.

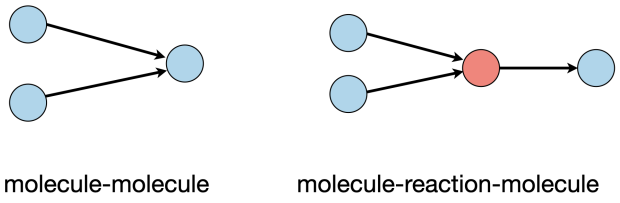


Figure 3.2: Graph representation of a reaction with two reactants and one product, using the different wiring schemes. Blue vertices represent molecules and the orange vertex represents a reaction.

3.3 Scale-free and Hierarchical Graphs

Barabási and Albert observed in 1999 that it is common for the vertex connectivity of large, real-world networks to follow a power law distribution and referred to this as scale-free behaviour [23]. The scale-free behaviour was found to be a result of a graph growing according to a preferential attachment mechanism. This means that new vertices are more likely to connect to already highly connected vertices. It was also suggested that this indicates a robust self-organizing phenomenon in a given graph. A visual example of a scale-free graph compared to a random graph can be seen in Figure 3.3.

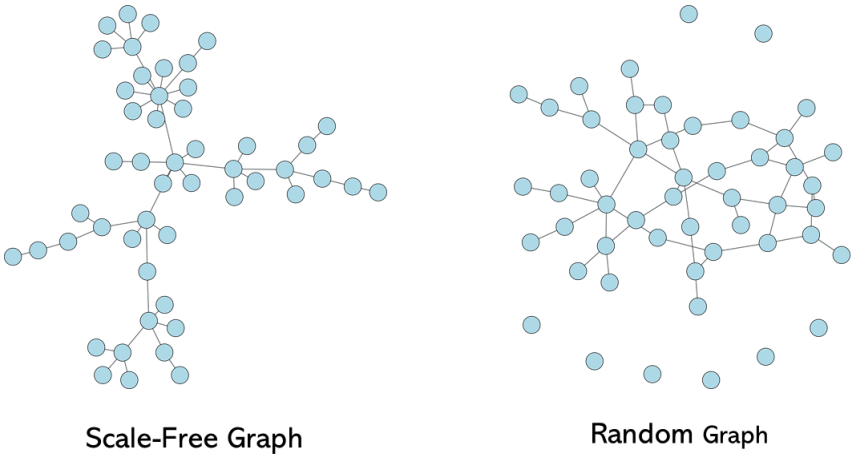


Figure 3.3: A scale-free graph compared to a random graph with the same number of vertices and edges.

In a scale-free graph, a small proportion of vertices have a significantly higher number of connections than the average, whereas the majority of vertices have few connections. In statistics this distribution is called a power law described by the scale invariant relation,

$$P(k) \propto k^{-\gamma} \tag{3.2}$$

where $P(k)$ is the probability to observe a vertex with degree k and γ is the power law exponent. The scale invariant property means that the distribution retains its features regardless of the scale of the variable. When considering directed graphs this can be with regards to either of the in-, out-, or total vertex degree. Power law is a heavy tailed distribution, meaning that the tail of the distribution is not exponentially bounded [24].

Scale-free graphs have, since their discovery, been a highly researched topic and many real-world graphs, such as the *World Wide Web*, biological graphs as well as chemical reaction graphs, [5, 6], have been found to approximately follow this distribution, typically with a power law exponent between 2 and 3. However, the term is somewhat controversial and some of these claims are being disputed. In many cases fitting the degree distribution to the log-normal distribution, another heavy-tailed distribution, is favourable compared to a power law which consequently means that the graph is not strictly scale-free according to [25].

The process of properly characterizing a scale-free graph comes down to two steps. First, the degree distribution is fitted to a power law and second, the resulting fit needs to be statistically evaluated. Ideally, the fit should be evaluated through a Kolmogorov-Smirnov goodness-of-fit test between the data and the estimated power law [26]. However, these tests often fail as any real-world graph is unlikely to be perfectly described by a power law. Statistical significance in these test also requires them to be run many times. This becomes computationally expensive for large graph sizes, due to the need for increasing number of iterations [24].

By reformulating the question to determine if power law is the better description of the data out of a set of distributions, the aforementioned issue can be avoided. The fit should then be compared to other candidate distributions by performing a likelihood-ratio test. Since the power law is a heavy tailed distribution, it should at minimum be compared to the exponential distribution. If the data is better described by the exponential distribution it consequently cannot be a heavy-tailed distribution, and thus is not scale-free. Furthermore, it should preferably also be compared to other heavy-tailed distributions such as log-normal [24].

3.3.1 Hubs

Highly connected vertices of a graph are called hubs and are present in scale-free networks. The *hubbiness* of a vertex is based on the degree centrality. Only the most connected vertices in a graph, with degree far above the average, are considered hubs [27].

The presence of hubs makes the graph more robust to vertex failure, meaning that

if a random vertex is removed from the graph it is much more likely to be one with a low degree, as these are more frequent in the graph. Removal of such vertex would not have a large impact on the graph structure. However, the presence of hubs makes the graph more vulnerable to adversarial, targeted attacks since it is sufficient to remove just a few hubs for the connectivity of the graph to break down. In chemical reaction graphs, hubs are usually intermediates in synthesis routes [6].

The presence of hubs, typically has the effect of reducing the average path length between all vertices in the graph, since vertices are more likely to be close to a hub. Related to this, a small-world network is defined as having an average path length, L, that scales proportionally with the logarithm of the number of vertices, N, according to

$$L \propto \log(N). \tag{3.3}$$

Typically small-world networks also have a high clustering coefficient [28]. This is not necessarily the case for scale-free networks which however commonly exhibit *small-world behaviour* because of the hub vertices that reduce the average shortest path length.

3.3.2 Betweenness Centrality

The hierarchical structure of a graph can be further explored via various vertex centrality metrics, describing a vertex's importance in some sense. The betweenness centrality of a vertex v is the number of shortest paths between all pairs of vertices v_s, v_t that runs through v, denoted $g_{s,t}(v)$, divided by all shortest paths $g_{s,t}$ between all pairs of vertices v_s, v_t, according to

$$b(v) = \sum_{s \neq t \neq v \in V} \frac{g_{s,t}(v)}{g_{s,t}}. \tag{3.4}$$

This metric is interpreted as a way to measure the traffic seen by a vertex. By further evaluating the average degree dependent betweenness centrality, one can get an approximation on whether there is a correlation between a vertex's centrality and its degree. If the betweenness increases with the degree, this indicates a hierarchical structure in the graph since highly connected vertices are also more important intermediates for synthesis paths. Figure 3.4 illustrates a graph where the red vertex has a particularly high betweenness centrality value.

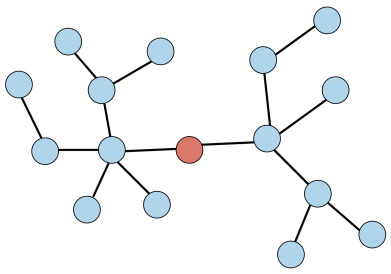


Figure 3.4: Example graph illustrating the betweenness centrality. The red vertex represents the one with the highest betweenness centrality in the graph, as many of all shortest path go through it.

3.3.3 Clustering

In some real-world networks it is typically the case that two neighbours of a vertex have an increased probability to also have a connection between themselves. For example, in a social network, if person A is friends with person B and person C, there is an increased probability that person B and C are friends as well. Clustering can be measured for the individual vertices of a graph or the whole graph at ones, called the local or global clustering coefficients respectively. Figure 3.5 illustrates a graph where the red vertex has a particularly high local clustering coefficient compared to the other vertices in the graph.

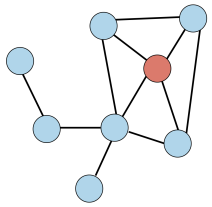


Figure 3.5: Example graph illustrating clustering. The red vertex has a particularly high local clustering coefficient, compared to the other vertices, as many of its neighbours are connected to each other.

The global clustering coefficient, C, is measured according to

$$C = \frac{6 \times \text{ Number of Triangles}}{\text{Number of Paths of Length Two}}. \tag{3.5}$$

Many real-world networks have been found to have global clustering coefficients between 0.01 and 0.5 [29]. On the contrary, reaction networks have been found to have low global clustering coefficients [6].

The local clustering coefficient, c_i, on the other hand measures the clustering for each vertex separately. It is defined as

$$c_i = \frac{2n_i}{k_i(k_i - 1)}, \tag{3.6}$$

for an undirected graph where n_i is the number of interconnections between the neighbors of vertex, i, and k_i is the degree of the same vertex. The denominator can be interpreted as the highest possible number of links between all neighbors of the given vertex.

As for the betweenness centrality, the local clustering coefficient dependence on vertex degree can be evaluated. This metric has been found to scale proportionally to k^{-1}, for some real-world networks, which also indicates a hierarchical structure in the network [6].

3.3.4 Graph Components

Another way to look at the hierarchy of a graph, often used in literature for organic networks in particular, is the presence of more and less connected components in the graph. Within graphs so-called Connected Components (CCs) can be identified, in which all pairs of vertices are connected via a path. In a directed graph the vertices of a CC only need to be connected in an undirected sense, i.e. there needs to exist a path between each pair of vertices only in one direction. A Strongly Connected Component (SCC) on the other hand, is defined as a sub-graph in directed graphs, such that each pair of vertices in the SCC has a path both ways between them. In a chemical sense this means that a synthesis route exists, in both directions, between each pair of molecules in this graph component [22].

Grzybowski et al. defined in [5] the largest SCC in a network of organic chemistry as the *core* of the network, but only if this component is of significantly larger size than all remaining ones in the graph. In the same publication, they defined the *periphery* of the network as the vertices which are not a part of the core but are connected to it. This corresponds to the CC for which the core component is a sub-graph. Molecules that have no connecting path to the core were defined as *islands*. These definitions are ambiguous and have been adopted in this study. The chemical reaction data analysed by Grzybowski et al. was extracted from the Beilstein database, in which they found the majority of molecules (78%) to be confined in the periphery, whilst 18% were found in islands and 4% made up the core. An analysis of the reaction graph architecture can help identify molecules that are central to chemical synthesis. Figure 3.6 illustrates an undirected graph with one hub, a central component and an island.

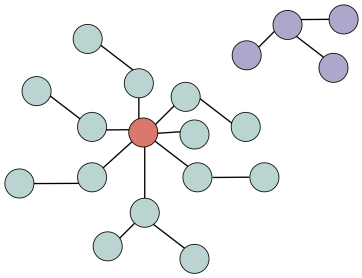


Figure 3.6: Example graph illustrating the graph components. As this is an undirected graph, the green vertices represent the CC, the purple ones form an island and the red vertex is a hub.

4

The Link Prediction Task

Traditionally, reactivity identification has been done through experiments [30], but a heuristic approach for link prediction in reaction graphs has also shown promising results for identifying novel chemical reactions [7]. Link prediction is defined as the task of predicting a relationship, a link, between two vertices in a graph. It is typically used for two different purposes; to infer missing or hidden links in an observed graph or to predict future links which could be added to the graph [31]. An example on the latter case could be to predict future friendships in social networks, predict protein-protein interactions in biological networks or as in this project, find novel chemical reactions.

The following sections outline the theory behind setting up a link prediction model in a knowledge graph. First, some heuristic approaches to link prediction are introduced in Section 4.1, followed by a description of the neural network approach used in this work in Section 4.2. The creation of the dataset is depicted in Section 4.3 and lastly, Section 4.4 outlines how link prediction performance can be evaluated.

4.1 Heuristic Approaches

Link prediction heuristics are simple yet effective methods for predicting links in a graph. These are convenient to use due to their simplicity and interpretability, however the drawback is that each heuristic comes with a strong assumption of when a link is more or less likely to occur.

One standard approach to heuristic link prediction involves so called *similarity-based algorithms*. In general, these algorithms compute the similarity of two vertices upon which the likelihood of a connecting link is determined. How this similarity is computed can vary significantly and the specific metric chosen can greatly affect the performance on a given graph [32]. A common practice is to distinguish between the order of these metrics. The similarity can for example be directly inferred from the intersection of neighbors between the two vertices, namely the number of Common Neighbors (CN) written

$$s_{i,j}^{\mathrm{CN}} = |\Gamma_i \cap \Gamma_j| \,, \tag{4.1}$$

for neighborhood Γ_i and Γ_i of vertices i and j. This is a first-order heuristic since it entails only one step of connections away from the vertices in question [33].

The similarity can also be measured by incorporating the union of the neighborhoods according to

$$s_{i,j}^{\text{JI}} = \frac{|\Gamma_i \cap \Gamma_j|}{|\Gamma_i \cup \Gamma_j|}. \tag{4.2}$$

As such this metric measures similarity and diversity simultaneously. This idea stems from research by Paul Jaccard [34], whom the metric is named after as Jaccard Index (JI).

An extension to the CN similarity is called Adamic-Adar (AA) [35]. This makes a distinction between the connectivity of neighbors, putting more weight on less connected vertices. The metric achieves this by summing up the inverse logarithmic degree centrality of common neighbors according to,

$$s_{i,j}^{\text{AA}} = \sum_{k \in \Gamma_i \cap \Gamma_j} \frac{1}{\log |\Gamma_k|}. \tag{4.3}$$

Because neighbors of neighbors are involved in this calculation, AA similarity is a second-order heuristic.

4.2 Deep Learning Approaches

Deep learning is a machine learning concept that allows a model to automatically learn a representation that can be used for detection or classification tasks. In contrast to the heuristics approaches to link prediction, the deep learning approaches are more complex and is not based on any initial assumptions. As described by LeCun et al. in [36], the general idea is to pass input data through stacked non-linear modules, which one by one learns more abstract representations. By using enough of these modules, very complex functions can be learned. This setup is referred to as a neural network which contains a set of learnable parameters, called *weights*.

When deep learning tasks are learnt using labeled data, it is called *supervised learning*. This is the case for the link prediction problem, where a potential edge, $e_{i,j}$, is labeled 1 if it is present in the graph and 0 if not. A data point, in this case a potential link, is passed through the neural network which outputs a probability of the link being present in the graph. If the ground truth of the specific link is 1 we want the probability to be as close to 1.0 as possible and vice versa. However, training is needed for the neural network model to achieve good such predictions.

As described by LeCun et al., the training of a neural network is an iterative optimization procedure that starts with *forward propagation*, which is one pass of the input data through the model to produce an output. The output is compared to the correct label and the *loss*, which is a measure on how far the prediction is from the ground truth, is calculated using an objective function, also called a *loss-function*.

The ultimate goal with the training is to minimize the loss, and this is typically done using the optimization algorithm *stochastic gradient descent*. In it, the gradients of the loss-function are derived with respect to the weights in the model and

backpropagated using the chain rule for derivatives. Starting from the final layer, the derivatives of the loss function are computed one layer at a time. This way redundant calculations can be avoided as the derivatives of each layer depend on the succeeding ones. The negative vector of the gradients points in the direction of the steepest descent and thus adjusting the weights in this direction takes the loss on average closer to a local minimum.

When this training process have been completed for all data points in the training set, this is called an *epoch*. After each epoch the model's performance is evaluated, on a separate validation set. Training is typically repeated for several epochs or until the validation performance has converged.

4.2.1 Graph Neural Networks

Due to the inherent properties and irregular structure of knowledge graphs, common deep learning architectures cannot directly make-use of this data. For this reason, so-called Graph Neural Networks (GNNs) were introduces in 2009 as an alternative deep learning technique [37].

Originally, GNNs were set up in a vertex classification context, namely predicting the labels of unseen vertices, we will see in Section 4.2.3 how it can be extended to link prediction. The general idea is to learn an embedding of a vertex by weighting the vertex's own feature vector together with the embeddings of its neighboring vertices. Features associated with edges can also be weighted into the vertex embeddings. How many orders of neighbours that are considered affects the depth of the GNN and thus the complexity of the GNN architecture grows with the depth. Typically, a small depth is preferred since it has been proven to be sufficient to look at a small proportion of the vertices at a time [38] and since more complex GNNs require more computations. An intuitive reason for this is that it would be redundant to include too many, or all vertices, for the embedding of a specific vertex.

The traditional training regimens described in the previous section, can then be applied to this special type of the network setup. Key ideas from many established variations of neural network have also been transferred to the graph domain. As an example, a graph convolutional network has been shown to outperform traditional GNN models by utilizing multiple convolutional filters and thus making the learning more efficient by reusing the learnable parameters [39]. Exploring and fitting reaction graph structure at different levels of detail for use in GNNs and measuring its performance is a challenging but promising avenue for research [40].

4.2.2 Graph Convolutions

The concept of convolutional filters in a GNN is non-trivial. Traditional Convolutional Neural Networks (CNNs) are usually applied on images where the convolutional filters can utilize the fixed order of the pixels. In contrast to most data structures graphs lack this natural ordering. Two approaches have been proposed to achieve graph convolution, spectral formulations relying on fixed graph structures

and spatial ones propagating local features between neighboring vertices.

Spatial graph convolution, first introduced in [39], takes a matrix of trainable parameters $\mathbf{W}$ and multiplies it with a vertex information matrix $\mathbf{X}$, the adjacency matrix including self-loops $\mathbf{A}$, and a diagonal matrix $\mathbf{D}$ containing vertex degrees. It then forward propagates this information according to,

$$\mathbf{Z} = f(\mathbf{D}^{\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}\mathbf{X}\mathbf{W}), \tag{4.4}$$

where f is a non-linear activation function. Multiple graph convolutions can later be achieved by propagating this activation matrix, $\mathbf{Z}$, through the succeeding layers according to

$$\mathbf{Z}^{t+1} = f(\mathbf{D}^{\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}\mathbf{Z}^{t}\mathbf{W}^{t}). \tag{4.5}$$

With Deep Graph Convolutional Neural Network (DGCNN), proposed in [41], it is possible to combine spatial graph convolutions with traditional CNNs by creating a consistent sorting of the vertices, called a SortPooling layer. In this model architecture the activation matrices from h consecutive graph convolutional layers are stacked on top of each other as $\mathbf{Z}^{1:h} = [\mathbf{Z}^1, ..., \mathbf{Z}^h]$ and used as input to the SortPooling layer. This layer utilizes the fact that graph convolutions have been shown to be a version of the Weisfeiler-Lehman (WL) algorithm, an algorithm used for graph isomorphism checking and graph similarity measurements. The SortPooling layer in DGCNN sorts the incoming information by using $\mathbf{Z}^h$, which contains the most refined WL signatures for each vertex and thus is consistent across graphs based on the graph topology. The same layer also adds a fixed pooling to the sorted data such that is size also becomes consistent. At this point, the data has become an appropriate input to traditional one-dimensional CNNs.

Another benefit of the SortPooling layer is that the gradients can be backpropagated through it, which allows the graph convolutions to be trained as well. The DGCNN architecture mixes a couple of one-dimensional convolutional layers with a MaxPooling layer and ends with two fully-connected layers as classification heads. A full explanatory schema of the DGCNN architecture can be seen in Figure 4.1. DGCNN was originally proposed for the graph classification task, but as we will see in Section 4.2.3 it can also be adjusted for the link prediction task using a dedicated algorithm called SEAL.

Graph convolutional layers can be tricky to train due to the high risk of overfitting. In [42] it was proposed that normalization through a vertex-wise variance-scaling operation could reduce such performance degradation. In particular they take the LayerNorm operation, originally used for traditional neural networks [43], and transfers it to the graph domain. As for any neural networks, another tool for reducing overfitting is to randomly cancel out a proportion of units and their connections in a given layer of the network by adding a dropout layer [44]. This is adapted on the last fully-connected layer of DGCNN, but can also be added to graph convolutional layers, as has been done in for example the Graph Attention Network (GAT), another graph convolutional model utilizing the attention mechanism [45].

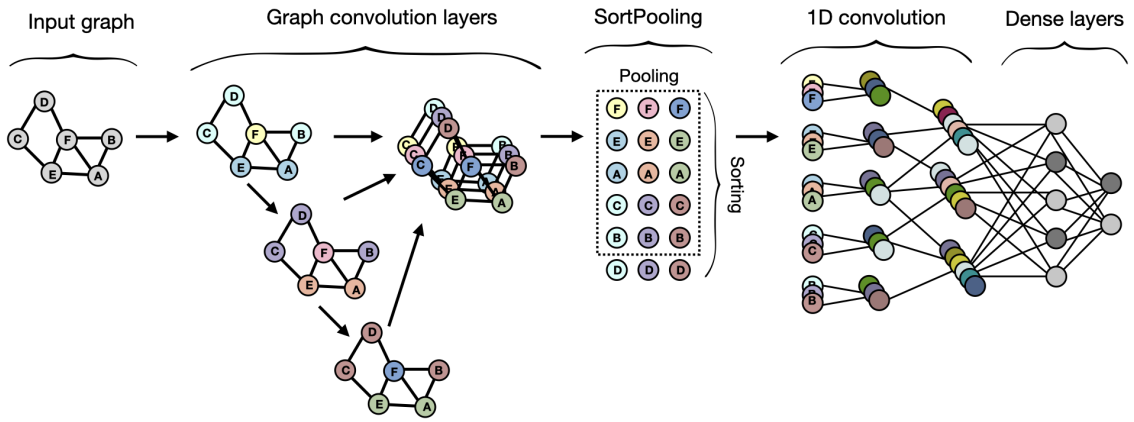


Figure 4.1: The DGCNN model architecture. Starting with an input graph, put through a number of graph convolutional layer refining the verticies' WL signatures which all get appended and passed to the SortPooling layer. After the vertices have been sorted they get flattened out to one dimension, pooled to a fixed size and passed through a number of convolutional layers before reaching a couple of fully-connected layers to perform the graph classification.

4.2.3 Learning from Sub-graphs

In graph classification tasks each data point is an entire graph, typically of relatively small size. For the link prediction tasks, one typically has a single large graph, because each data instance is rather two vertices and the potential of a link between them. Training such a model on the entire graph at once for each link would be infeasible, instead it has been shown that it suffices to use a sub-graph of neighboring vertices to learn meaningful representations [38]. Thereupon a link prediction algorithm called SEAL was introduced, for extracting such sub-graphs and creating a labeling which effectively indicates a vertex's structural impact on the source and target of a given link. SEAL stands for *learning from Sub-graphs, Embeddings and Attributes for Link prediction* and the method has outperformed other state-of-the-art models, such as Variational Graph Auto-Encoders (VGAE) [46], on the Open Graph Benchmark (OGB) [47] datasets for link prediction [48].

The SEAL algorithm can be summarized in the following three steps.

1. Extraction of a k-hop enclosing sub-graph adjacency list, $\mathbf{A}$, where k is the depth of neighbors included in the sub-graph.

2. Construction of a vertex information matrix, $\mathbf{X}$, for the given sub-graph with vertex feature vectors as rows which can include vertex labellings, embeddings and/or attributes.

3. Forward propagation of $(\mathbf{A}, \mathbf{X})$ through a GNN.

The most important part of the vertex information matrix is the labeling. This can

for example be done using the Double Radius Node Labelling (DRNL) strategy, another contribution from [38], which gives each vertex in a given sub-graph an integer label based on its relation to the vertices of the potential link. Most importantly this labeling highlights the source and target vertices such that ultimately the model can separate them from the rest. Apart from this, vertices are labeled based on their radial distance to both of the vertices of the given link such that it reflects their relative position and structural importance. Figure 4.2 illustrates the three steps of the SEAL algorithm.

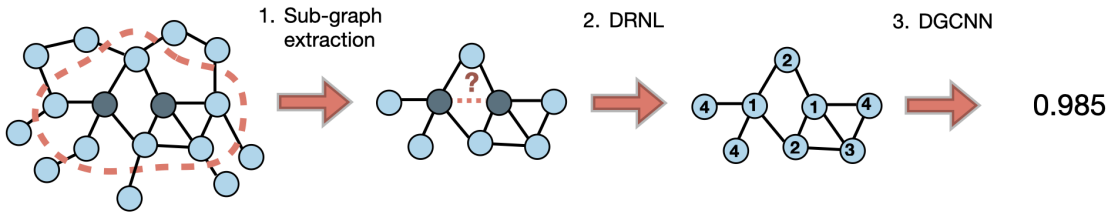


Figure 4.2: End-to-end SEAL algorithm. 1. Extracting a 1-hop sub-graph around the darker source and target vertices. 2. Vertex labelling according to the DRNL method. 3. Forward pass through DGCNN model for predicted likelihood of link.

Apart from the vertex labelling, SEAL also allows for additional information being appended to the $\mathbf{X}$ matrix, as stated above. This can be more elaborate embeddings of the sub-graph structure and topology calculated by some algorithm, such as Node2Vec [49], or model, such as VGAE [46]. It can also be information known about what the vertices represent. In the context of chemical reactions, the Molecular Fingerprints are suitable such attributes. They are promising since they are unique bit vectors that can be computed from molecular SMILES as well as SMARTS (e.g. templates). The inclusion of attributes typically has a positive impact on the training of the SEAL algorithm, but embeddings was found to sometimes hurt the performance by the original creators.

Any GNN model can be used for the final step of SEAL but in the original paper DGCNN was proposed. Note that, due to the DRNL algorithm making a clear distinction between source and target vertices, the link prediction is equivalent to graph classification of the sub-graph.

4.3 Sampling of the Negative Class

Link prediction is in its essence a binary classification problem where the task is to predict *link* or *no-link*. However, real-world graphs are usually large and sparse resulting in highly imbalanced classes where the positive class (link) is usually only a small fraction of the negative class (no link). An instance of the negative class is referred to as a *negative link*, while in contrast, an actual link is referred to as a *positive link*. As it is infeasible to train on all negative links, [38] instead proposed to sample as many negative links as there are positive ones, giving the illusion of a

balanced training set. This requires a method for selecting which instances of the negative class should be included, from here on referred to as *negative sampling*.

In [38] the negatives are sampled randomly, such that source and target vertices of each negative link is sampled randomly from all vertices in the graph. There are many alternative ways of doing this so that the negative class is more similar to the positive class. The choice of sampling strategy is likely to affects the average path lengths between the sampled source and target vertices, which in turn affects the predictions made by the network. This is important because it has been shown that the predicted probability of a link tends to increase with a shorter distance between the source and target vertices [50]. The assumption that different negative sampling strategies generates "harder" or "easier" negatives is partly inspired by the reasoning behind negative sampling in [51]. Further, as SEAL extracts a k-hop sub-graph around the target link, as described in Section 4.2.3, longer distances between the source and target vertices result in more disconnected sub-graphs, as illustrated in Figure 4.3. Regardless, sampling of the negative class is likely to impact the model's performance and its ability to generalize, and consequently needs to be carefully considered.

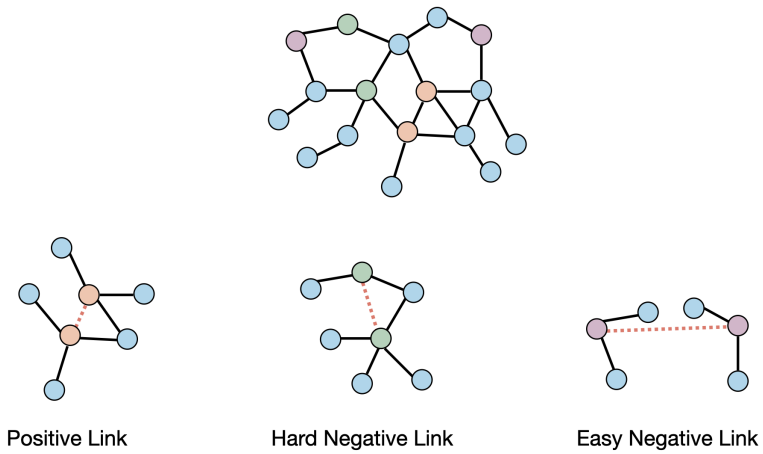


Figure 4.3: Example graph for illustrating the effect of negative sampling when using SEAL. Showing a positive, an "easy" negative link and a "hard" negative link, which more closely resembles a positive link. The colored nodes in the graph above is to mark the node pair of each sampled link, and the orange dotted line represents the target link.

4.4 Performance Metrics

Standard metrics for binary classification can be divided into fixed threshold metrics and threshold curves. Accuracy is an example of a fixed threshold metric. The drawback with these kinds of metrics is that the threshold for separating the classes needs to be predefined which might not be straightforward. In binary classification the terms True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) denote the number of predictions which either correctly classifies an

example as the positive vs negative class or incorrectly classifies it at such, at a given threshold.

An example of a metric which does not use threshold predictions at a fixed level, is the Receiver Operating Characteristic (ROC)-curve. This curve shows the relation between TP Rate (TPR) and FP Rate (FPR), defined as,

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{4.6}$$

For random guessing the ROC-curve approaches the identity function and from this baseline, curves closer to the upper left corner of the plot indicates better predictions. This is a convenient way to evaluate individual results, however when comparing different models it is better to have a single measurement. For this purpose the Area Under the ROC-curve (AUC) is most commonly used, ranging from 0 to 1 with highest score for perfect classification and 0.5 for the random case. Another advantage of the AUC is that it is *scale-invariant* as it measures the rankings of the predictions rather that their absolute values.

The two metrics *precision* and *recall*, defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.7}$$

can similarly be used to draw the so-called *precision-recall curve*. Taking the area under this curve is called the Average Precision (AP) and is another way to effectively evaluate binary classification.

Specialized metrics have also been created for the link prediction task, incorporating the fact that the negative class is not necessarily as "correct" as the positive class. One such metric is the Hits@K score [48], originally meant to evaluate recommendation systems where it is also the case that the positive class is more certain than the negative class. The score counts how many of the instances in the predicted positive class are predicted as positives with higher certainty in comparison to the negative class. More specifically, it takes the ratio of positive edges ranked at K-th place or above against all negative edges.

A distinction can be made between the metrics described above, which weigh all vertices equally calling, them *link-equity* metrics, and ones that do not *vertex-equity* metrics [52]. The latter have been proposed to be more robust to biased performance on hub vertices, for which links are typically easier for link prediction models to predict. In general, a vertex-equity metric can be created by singling out all links with a given vertex as source or target and computing a link-equity metric for them. This is then done for all vertices and averaged to get the final measurement. AP has been used in this way, referred to as the Mean Average Precision (MAP) [53].

In traditional classification tasks, it is typically enough to look at the evaluation metrics compared to some baseline in order to get a fair assessment of the model's performance and ability to generalize. This is because labels in the test set are typically known to be true. It should also be considered that in some link prediction problems, the negative class might correspond to missing or future links.

5

Methods

As a means to better understand the reaction graph and the chemistry performed at AstraZeneca, a thorough analysis of the reaction graph was performed, including a comparison between a subset of the internal ELN, hereby simply referred to as ELN, and the public reaction graph, USPTO. After the analysis, the ELN reaction graph was further explored with link prediction to infer missing links. To allow for a fair comparison between the internal reaction data and the public dataset, ELN and USPTO have previously been set-up in a shared graph architecture by AstraZeneca researchers.

This chapter is essentially divided into three sections. In Section 5.1 the graph architecture is explained, followed by Sections 5.2 and 5.3 which describe the methods used for the two parts of the project; Graph Analysis and Link Prediction.

5.1 Knowledge Graph Architecture

The reaction knowledge graph supports numerous purposes within AstraZeneca including different research topics and for this reason its selected architecture is rather complex. Figure 5.1 shows an overview of the graph architecture with the four vertex types that were central for this project, and how they are connected; *molecule*, *reaction*, *reaction template* and *molecule template*.

Molecule vertices represent the reactant and product molecules involved in a reaction, which in turn is encoded as a separate reaction vertex. These vertex types are connected so that all reactants from a specific reaction are connected to that reaction vertex, which accordingly is connected to the product molecule of the reaction. Each reaction vertex is additionally connected to a vertex of a different type, a reaction template vertex. More than one reaction can be connected to the same reaction template since the reaction template is a more general description of the reaction, as described in Section 2.1.

Reaction template vertices are also connected to molecule template vertices corresponding to the individual templates of reactants. All vertices corresponding to templates, both for reactions and molecules, have a radius property related to the depth around a reaction site that is included in the template, as described in Section 2.1. In addition, the molecule template vertices have an interlinkage between

themselves, shown as a self-connection in Figure 5.1. These links represent a *reacts* relation between them, i.e. that two molecule templates vertices links to the same reaction template vertex. This link is a presumption for the link prediction as it was implemented in this project, and is the only link where the direction lacks meaning. The complete graph includes additional information related to reactions and its metadata, but this was omitted from Figure 5.1 for simplicity.

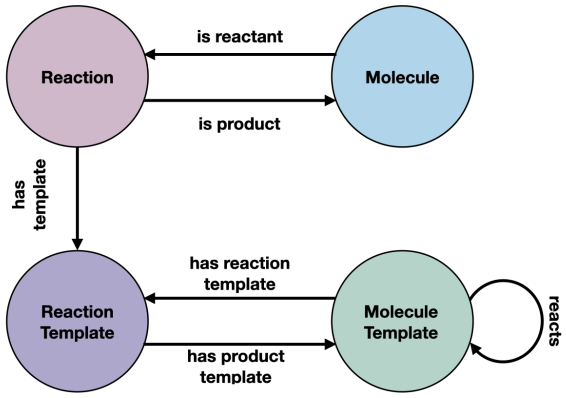


Figure 5.1: A simplified schema of the knowledge graph used in this project, showing the relation between different types of vertices. Edges are distinguished from each other by the labels written in the figure. Both data from AstraZeneca's internal graph, ELN, and the public USPTO graph are contained in the same database.

The entire graph is encoded using Neo4j, a graph database management system. Thus, management of this graph was done using Cypher queries through the Neo4j driver for Python, such that relevant parts could be extracted to GraphML files. This format enables the data to be efficiently imported into graph-tool [54], which was the main library used to represent the graph in Python and for performing the graph analysis. There exists many libraries for handling graphs in Python, but specifically graph-tool was used because of its extensive set of graph algorithms implemented and prepared for parallelization, convenient for handling large datasets. RDKit is a cheminformatic toolkit [55], that was used for all tasks related to handling chemical structures and templates. For deep learning related tasks regarding the link prediction PyTorch was used together with PyTorch-geometric, an extension of PyTorch for irregular data structures [56].

5.2 Graph Analysis

Initially, statistical properties of the internal ELN graph and the publicly available USPTO graph was analyzed and compared. This analysis was done on a subset of the graph that only contained the molecule and reaction vertices and gave valuable insights to the graphs' behaviour as well as point to chemical interpretations of the graph structure. Specifically, this study focused on investigating whether the graphs exhibit similar properties as have been found in published literature [5, 6].

The overall goal was centered around evaluating if the respective graphs are scale-free or exhibit typical behaviours thereof. A graph hierarchy is one such behaviour, others are the presence of hubs and the small-world property.

For the purpose of measuring different graph metrics, the bipartite graph schema, with both molecule and reaction vertices, was in some cases compared to a schema with only molecule vertices. The choice depended on the specific question at hand as well as previous work for comparison purposes. The molecule-molecule schema was a reconstruction of the bipartite graph where the links were rewired such that, for every reaction vertex, all in-coming neighbors were connected to all of its out-going neighbors, followed by the removal of the reaction vertex as well as any parallel edges. This resulted in a graph with an *all-to-all* wiring, meaning that all reactant molecules connected to all product molecules of a reaction.

In order to mitigate the risk of a potential "size effect" when comparing the knowledge graphs based on these two reaction databases, a subset of USPTO was extracted of similar size to ELN. Only chemistry for reactions registered before 1998 was included in this subset. We believe that this size reduction approach is appropriated to decrease the risk of bias when selecting the reactions being part of the smaller graph.

5.2.1 Scale-Free and Hierarchical Graphs

As previously stated, the process of determining if a graph is scale-free, comes down to fitting the degree distributions to a power law. There are different ways of doing this, both regarding which of the in-, out- or total degrees are used and regarding which vertices to consider, all vertices or the different types of vertices separately. Depending on exactly what data is used, the interpretation of the result can differ. The total degree on all vertices mostly gives a general overview of the graph whilst for example the out-degree of molecule vertices says something about the number of reactions that molecules are involved in and so on. Thus, multiple variations of the data were used in this step.

Fit to Power Law Each fit was done towards a discrete power law function using the python package `powerlaw` [24]. In this funcion the parameter $k_{\min}$ determines the lowest degree to draw samples from, and it was automatically estimated to give the best possible fit. This fit was then compared to ones between the empirical distribution and other distributions, namely log-normal, log-normal positive, truncated power law, exponential, and stretched exponential. The same package provides support for this comparison and the results are given in terms of a likelihood-ratio, R, between power law and the respective alternative distribution. If the data is more likely to have been generated by the first distribution, R takes positive values and if the second distribution is more probable R is negative [24]. The p-value indicates the significance of the R-value. If this value is above a certain threshold, suggestively above 0.05-0.1, neither of the distributions are favoured since the direction of R is likely affected by fluctuations in the data [6].

Betweenness centrality and Clustering Since hierarchy is common for scale-free graphs, this was also explored. Such tendencies can be found in different ways. The betweenness centrality gives an indication of which vertices are of higher importance, which in this context means that many synthesis routes run through a given vertex [6]. Graph-tool directly provides a function for computing the betweenness. The analysis of the average degree dependent betweenness, plotted on a log-log scale leads to a more interesting observation. An upward such trend directly indicates that more connected vertices have a higher betweenness, which implies a hierarchical structure between the vertices. The average betweenness centrality of each degree, k, was calculated according to,

$$\bar{b}(k) = \frac{1}{N_k} \sum_{v \in V} b(v) \delta_{k,k_v}, \tag{5.1}$$

where δ_{k,k_v} is the Kronecker delta function cancelling out all terms for which the degree of vertex v is not equal to k. The average is taken over the number of vertices of degree k, N_k.

Another way to see indications of a hierarchical structure in a graph is through the global and local clustering coefficients. Since the complete, bipartite graph by definition cannot be clustered and thus has clustering coefficients exactly equal to zero, these metrics were only calculated for the molecule graph. Same as for betweenness centrality, the local clustering coefficients can be evaluated in terms of vertex degree in order find a hierarchical structure. Similarly, this is referred to as the average degree dependent clustering, $\bar{c}(k)$. For some real-world, scale-free networks the average degree dependent clustering coefficient have been found to scale proportionally with k^{-1}. Similarly as for betweenness, the average degree dependent clustering coefficient was therefore calculated as,

$$\bar{c}(k) = \frac{1}{N_k} \sum_{i=1}^{N} c_i \delta_{k_i,k}, \tag{5.2}$$

where N_k is the number of vertices with degree k and $\delta_{k_i,k}$ is the Kronecker delta that cancels all clustering coefficients not equal to k.

Hubs The next step of the analysis was to identify and characterize *hub molecules*. In practice this required choosing a threshold on the degree centrality, above which vertices were considered hubs. Finding these hubs served two purposes, one regarding their effect on the graph and one in relation to their chemical significance. On the first account, hubs tend to shorten the path lengths in a graph, often resulting in a small-world behaviour. From a chemical point of view, the presence of hubs indicates that any substance may be synthesised in, on average, only a small number of reaction steps from any other substance in the graph. To evaluate this the `distance_histogram` function in graph-tool was used which yields a histogram of each measured distance, and can be used to efficiently find the average shortest path length. Hubs are also presumed to be essential building-blocks for the chemistry in the graph. This was corroborated by looking at a number of molecular properties.

Network Components In search for chemically meaningful components of the reaction graphs, community structures were analyzed. Firstly, the presence of a core was evaluated in each graph, after which the periphery and island component were identified. Built-in methods in the graph-tool package were used to find both SCCs and ones connected in an undirected sense, i.e. CCs. Their sizes were then compared, using a histogram, such that if a single SCC had significantly larger size than the remaining ones, this was identified as the core of the graph. On the other hand, if no such SCC could be determined, the graph did not present a core. For identifying the periphery and island components the CCs were found. If a core existed, the CC containing the core was chosen as the periphery, excluding the core vertices, otherwise the largest CC was chosen as the periphery. Lastly, all remaining CCs were labeled as islands. The resulting proportions of each structure present in each graph were compared to the proportions found in literature [22].

For the purpose of exploring the importance of different molecular descriptors, a number of them were also computed for each molecule vertex and compared between the network structure component, hubs as well as between the two graphs. This helped to answer questions such as whether core molecules are the least complex ones, i.e. building-block substances, which lay ground to most of the remaining chemistry in the graph as was found in literature [22]. And similarly, if their complexity increases by moving out to the periphery leaving the most complex ones in the islands.

5.3 Development of the Link Prediction Model

After the graphs had been analyzed, the project focused on the development and training of a link prediction model to suggest novel reactions and new chemistry. This was done through an optimization procedure and numerous evaluation steps.

It was not trivial how link prediction could be applied to the bipartite graph of molecule and reaction vertices. The reason for this is because prediction of a new reaction in this case would actually include several links and the creation of a new reaction vertex. This was the main motivation for transferring the bipartite graph setup to a monopartite version presenting only reactant molecules with interlinkage indicating a reacts relation. Several reasons were behind the decision to do predictions on the molecular template level of the graph. For one this made it a smaller graph, hence computationally easier to train on, and also it allowed for a more generalized model since the relations themselves were more general.

As previously stated, this task was performed only on the ELN graph data. The following sections describe how the data was processed, the implementation of the model, its training, and the various evaluation methods used.

5.3.1 Data Processing

Before the graph could be used as input to the SEAL algorithm a number of preprocessing steps had to be performed. First of all, the molecule template vertices

from the ELN source had to be extracted together with their internal links from the Neo4j database. This was done separately for templates of radius 1 and 2 from reaction sites. The Morgan fingerprint, of size 1024 and radius 2, was then generated for every molecule template, such that these could be added as vertex attributes. These calculations were done using internal modules at AstraZeneca which were based on RDKit functions.

Splitting Strategies All reactions have a creation date property, spanning from 2003-2019. The edges were split into different sets for training, cross-validation, and testing using this information derived to each reacts relation in the molecule template sub-graph. Since the date property was connected to the reaction rather than to the reaction template, each reaction template could be associated with several date properties. To address this a conservative approach was taken, by associating the earliest date to each reaction template, and reacts relationship, as this indicated the creation of that specific template. The number of edges from each year varies, but Table 5.1 present a reasonable split between the years such that each fold contains similar amounts. As the ultimate goal of the link prediction relates to future findings in the graph, the best option for a test set was chosen to be the last few years of data, i.e. links from 2017-2019.

	2003-2006	2007	2008	2009	2010	2011	2012	2013	2014	2015-2016	2017-2019
Radius 1	496	2469	2374	2067	1685	1512	956	1168	1301	1704	1542
Radius 2	598	4304	5088	4919	3963	3307	2258	2939	3365	4398	3870

Table 5.1: Number of new reacts links added to the ELN molecule template graph of radius 1 and 2 per year. A few years have been combined to create a somewhat even distribution between folds.

Links from before 2017 were split into ten folds in two different ways, randomly and based on the year of entry. The randomly generated folds were used to perform cross-validation in the hyperparameter search, alternating the validation set among the ten folds. The time dependent folds were taken according to the split in Table 5.1 and were used to show how the model's performance were affected by the size of the training data. In this case, training was first done on only the first fold, containing links from 2003-2006, and validated on the succeeding fold, year 2007. Next the first two folds, i.e. links from 2003-2007 were used in for training while the links from 2008 were used for validation. This incremental process was pursued until the last case, of training on years 2003-2014 and validating on the last fold, excluding the testing one, containing links from the years 2015-2016.

Negative Sampling Three different strategies were used to sample negatives for the training and evaluation sets; one random, one from the positive links' vertex distribution and one being a mix of both. The hypothesis, which was put to the test as part of this work, was that a randomly sampled negative link on average has a longer distance between its vertices than for the alternative sampling strategies.

Random negative sampling refers to sampling the negative class at random, meaning that both vertices of the negative link were uniformly drawn from vertices of all positive links in a given set. On the other hand, *negative sampling from distribution* is when the negative class was sampled from the same distribution as the positive class. The motivation behind this method was to have a negative class more similar to the positive class than what was obtained by random negative sampling. In this strategy, the sampling of source and target vertices for the negative class was done with replacement from the source and target vertices of all positive links. Finally, what is referred to as *structured negative sampling* is a mixture of random sampling and sampling from distribution. The source vertex was sampled from the distribution of the positive link's vertices (with replacement) whereas the target vertex was sampled uniformly from all vertices of positive links in the given set.

After all negative links had been sampled, the resulting negative links had to be checked for overlap with the positive class in order to confirm that no negative links were also positive. Such errors were corrected through re-sampling until no overlaps remained Note that each strategy described above, was done with respect to the positive links in each of the train/validation/test sets separately. Thus the distributions referred to were only the ones from the given data split.

Oversampling of the Negative Class As the number of possible negative links in the graph greatly exceeds the number of positive links, sampling of the negative class was performed as explained above. Because this sampling of the negative class restricts the model to only a small fraction of the negative class an oversampling option was implemented, with the motivation that this would expose the network to a larger variation of the negative class. The oversampling method initially samples more negative links than positive but during training it selects a random subset of all the sampled negative links, equally many as the positive links. In this way the model encounters a larger variety of the negative links while still training on a seemingly balanced dataset. The implementation required a factor for how many times larger the negative set should be compared to the positive set.

5.3.2 Model Implementation and Training

An open source implementation of SEAL [57] is available using PyTorch, with the extension library PyTorch Geometric for handling irregular data structures such as graphs. This includes functions for the sub-graph extraction, the vertex labelling with DRNL and a version of the DGCNN model as described in Section 4.2.1. The end-to-end model for this project was largely based on this implementation but adapted to the ELN molecule template graph. Apart from the necessary adaptations, some of the original functionality was simplified to more easily suit the given application and much of the training procedure was rewritten for easier understanding and use.

Improvements were also attempted to the DGCNN model, including the addition of layer normalization to the graph convolutional, batch normalization as well as dropout between the one-dimensional convolutional layers. An exponential sched-

uler was introduced for the learning rate such that the learning rate decreased with decay rate, $\gamma = 0.9$, each epoch. The model was trained to optimize the Binary Cross-Entropy (BCE) loss function, using `BCEWithLogitsLoss` in PyTorch, with the Adam optimizer as stochastic gradient decent algorithm [58]. As in the original implementation, hyperbolic tangent was used as activation for all graph convolutional layer whereas the commonly used Rectified Linear Unit (ReLU) activation was used for all other convolutional and fully-connected layers apart from the last layer. Since the `BCEWithLogitsLoss` has a built-in Sigmoid layer, no activation was applied to the last layer. Instead the final predictions were manually passed through a Sigmoid function whenever needed for evaluation of predictions.

Otherwise, a standard training procedure was used, as explained in detail in Algorithm 1. The outer loop in the algorithm refers to re-running training on new data, either on only new sampled negatives or on a different training set all together as for the cross-validation. Apart from this, other evaluation metrics were additionally computed from each epoch of each run, both concerning training and validation performance. This included AUC, based on which the best model from each run was also checkpointed for future use and evaluation.

Algorithm 1
Training procedure of SEAL algorithm

```
Import graph, G(V, E)
for all runs do
    Create predefined data split of edges E
    Sample set of negative edges, Ē
    Initialize dataloaders with sub-graph extraction and vertex labelling
    Initialize DGCNN parameters w ~ U(0, 1) and learning rate, α
    for all epochs do
        for all training batches do
            Reset optimizer gradients
            Forward pass through model
            Compute loss
            Backpropagate gradients
            Update parameters
        end for
        for all validation batches do
            Forward pass through model
            Compute loss
            if best performance then
                Checkpoint model
            end if
        end for
        Decrease α according to α = γα
    end for
end for
```

Hyperparameter search Due to the large space of hyperparameters and settings of the training procedure, all possible combinations could not be evaluated. Instead a reasonable starting point was decided from which a cross-validation was performed between scenarios where a single configuration was changed at a time from the standard settings. Table 5.2 presents the standard settings as well as the different alternatives explored.

	#Hops	#Hidden Channels	Dropout	Normalization	Attributes	Embeddings
Standard	2	8	0.5	False	Fingerprints	False
Hop 1	**1**	8	0.5	False	Fingerprints	False
Hop 3	**3**	8	0.5	False	Fingerprints	False
Hidden Channels 16	2	**16**	0.5	False	Fingerprints	False
Hidden Channels 32	2	**32**	0.5	False	Fingerprints	False
Hidden Channels 64	2	**64**	0.5	False	Fingerprints	False
No Dropout	2	8	**None**	False	Fingerprints	False
Normalization	2	8	0.5	**True**	Fingerprints	False
No Attributes	2	8	0.5	False	**None**	False
Embeddings	2	8	0.5	False	Fingerprints	**True**

Table 5.2: Standard hyperparameters chosen, with alternatives compared through cross-validation. *Normalization* refers to LayerNorm between graph convolutional layers and batch-normalization between remaining convolutional layers. *Embeddings* refers to an added layer to input of the model, for computing vertex embeddings.

The best model performance in terms of AUC was averaged over all ten cross-validation folds and the configuration which achieved highest overall score was chosen going forward. Note that this search method relies heavily on the assumption that the hyperparameters are independent from each other, which is not necessarily the case and thus a better optimum could likely be achieved with a more thorough search.

Furthermore, the hyperparameter search was only performed for the random negative sampling method and the ELN graph with molecule templates of radius 1. The optimal configurations from this case were then used for comparison between the other negative sampling methods, the oversampling strategy as well as tests made on the graph with molecule templates of radius 2. Also here, there could be differences in optimal settings such that the chosen method resulted in a biased advantage to the original case.

Another interesting aspect explored was the size of the training data. It was expected that, at some point, when decreasing the size of the training data the model performance should collapse. The reason for this is overfitting, i.e. that the included information is not sufficiently diverse such that generalizability is achievable. For this the time splits from Table 5.1 were used, as explained in Section 5.3.1. This test was especially useful when comparing the graph versions with different molecule template radius, since this graph's size varies significantly. These results could be used to compare the model's performance trained on the two variants but of similar size.

5.3.3 Traditional Evaluation of Binary Classifier

All trained models were evaluated on the same test set, as described previously. Apart from the positive links, this set also contained a fixed set of negative links sampled once in advance. These were sampled from all three negative sampling strategies, resulting in three times as many negative links as positive links. The reason for this was to enable evaluation of the impact that each negative sampling method had on what the model had actually learned. Ultimately, this test set was used as three separate sets each of which containing the same positive links but different negatives.

The resulting predictions on the test sets were then used to draw ROC-curves and compute the AUC. Further, the distributions of predictions for links from each of the negative sampling methods as well as the positives were analyzed. This was a way to visualize the confidence of each model. Apart from this, a metric more specialized for link prediction was also computed, namely the Hits@K score described in Section 4.4 for $K = 20, 50, 100$. With scalar model outputs defined as y^+ and y^- for predictions on ground truth positive edges and ground truth negative edges respectively, the Hits@K score was computed as follows,

$$\text{Hits@}K = \sum_{y \,:\, y^+ > y_K^-} \frac{y}{n}. \tag{5.3}$$

This means that all prediction scores on positive edges that are higher than the top K highest prediction score of a negative link, i.e. y_K^-, gets summed together and divided by the number of ground truth positive edges, n.

Next, the evaluation explored the vertex-equity metric MAP for the purpose of abstracting inequalities in vertex connectivity which were believed to affect the previous link-equity metrics to some extent. An optimal threshold, θ, was also chosen such that,

$$\text{TPR}(\theta) = 1 - \text{FPR}(\theta), \tag{5.4}$$

held approximately such that TPR was as high as possible whilst also minimizing FPR. With this threshold applied additional metrics were computed. All of these metrics were then also calculated for predictions by the heuristic baseline models mentioned in Section 4.1, namely CN, AA similarity, and JI.

5.3.4 Additional Evaluation of Predicted Reactions

A missing link in the internal reaction graph does not necessarily mean that the two substances cannot react, but instead reflects that the reaction has not been successfully performed, or even explored, at AstraZeneca. This means that the previously discussed evaluation metrics could be misleading. As the ultimate goal with link prediction was to predict future links in the given graph, it was of great importance to also evaluate the chemical validity of the predicted reactions. Thus, apart from the evaluation metrics, two additional steps were made to check the plausibility of the predicted reactions; a search in USPTO and generation of reaction templates.

Search in USPTO It was possible to identify some of the missing links in the ground truth of the graph through a search in USPTO, as there were some overlaps of molecules between the two graphs. This search was used to check if the negative links in ELN were also present in USPTO. Such a link directly represented a known reaction outside of AstraZeneca's reaction data. As the model was trained solely on ELN, a high prediction score on these links could be interpreted as a correctly identified missing link. However, note that USPTO does not include all possible chemical reactions either, and consequently this search was not complete.

Another aspect of this is that a large number of the molecules included in the graph could in theory react. This does not mean that all of those additional links are of interest to include in the internal graph at AstraZeneca, as their graph context is pharmaceutical drug discovery and synthesis.

Generation of Reaction Templates An additional step to the evaluation was to look at whether or not the the predicted molecule template pairs could be combined to reaction templates. Molecule templates have half-templates associated to them, via its association with reaction template. Each half-template includes a reactant's transformation to the product, including the mapping of atoms. Note that the set of half-templates, associated to each molecule template, are a result of what reaction templates the molecule template participates in. How these parts are connected is illustrated in Figure 5.2.

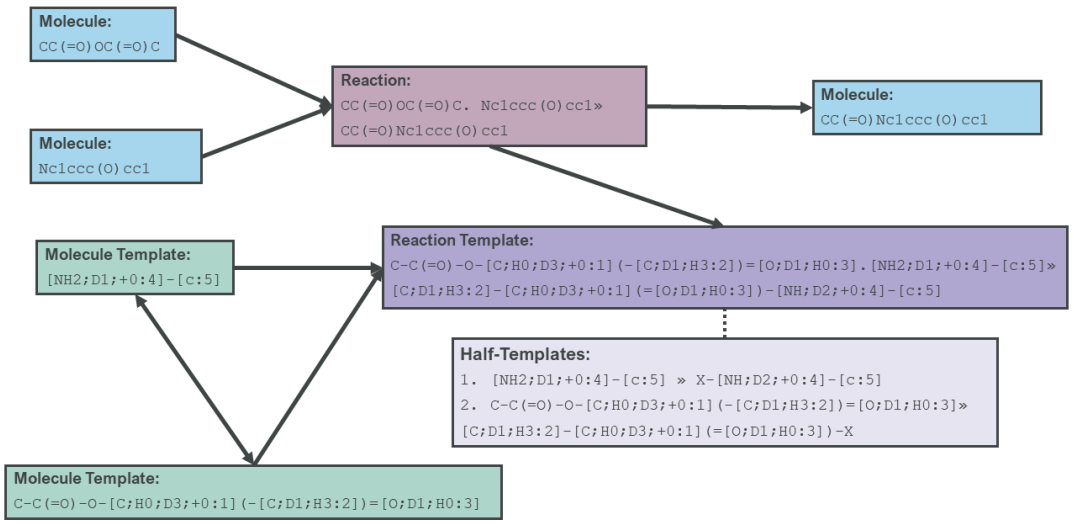


Figure 5.2: Schema of the paracetamol reaction presented Figure 2.2. Blue boxes contain SMILES for each molecule and the pink box holds the reaction SMILES. The purple box contains the reaction template and in the green boxes are the molecule templates. The lighter-purple box shows the half-templates associated with the reaction template. Note, the product molecule template has been omitted from this schema for simplicity, since it is not important for the implementation.

Consequently, if two molecule templates can be combined and create the left-hand side of a reaction template, i.e. if the half-templates are compatible, a new reaction template can be generated. This was checked by attempting to combine any pair of half-templates which were associated with the molecule templates to create the reaction template. As the creation of molecule templates into reaction template were dependent on what half-templates were present in the graph, failure to create a reaction template did not necessarily mean that a reaction template could not be made but rather that no similar reaction was present in the graph. If such a reaction template could be created it meant that a reaction could be possible.

6

Graph Analysis

Throughout this section, the results from the analysis of the bipartite reaction graphs ELN and USPTO graphs are presented and discussed. ELN has 389 123 molecule and reaction vertices while USPTO is considerably larger with 2 457 226 such vertices. Results for alternative graph versions have also been added for comparison when needed, including the rewired molecule graphs and the subset of USPTO snapshotted in the year 1998, with 405 297 vertices. Since the link prediction, in the second part of this project, was done on the molecule templates part of ELN all results shown in this sections that are relevant on that graph version, for both radius 1 and 2, can be found in Appendix B. Those results can be seen to be reasonably similar to the ones shown here, such that similar conclusions can be drawn.

6.1 The Scale-free Property

Firstly, the degree distributions for ELN and USPTO, shown as histograms in Figure 6.1, were fitted to power law as described in Section 5.2. The degree histograms show that many vertices have only few connections while fewer ones are highly connected, similarly to what was previous reported [6].

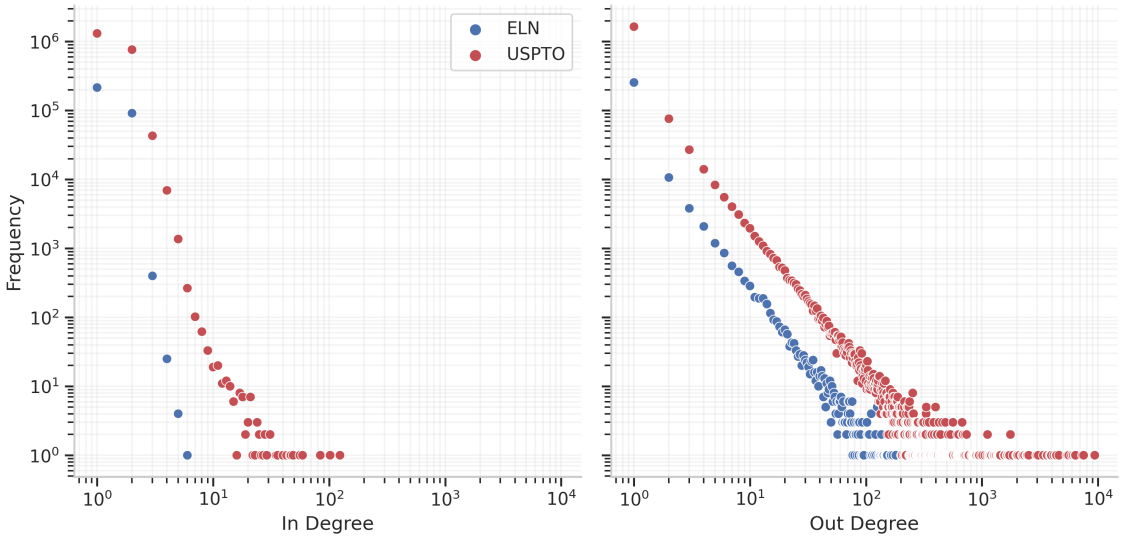


Figure 6.1: Degree histograms for in- and out-degree of the two graphs.

The process of fitting the degree distributions to a power law was done for ELN and USPTO regarding all three degree types. These fits resulted in the estimated parameters presented in Table 6.1, where γ is the power law exponent and $k_{\min}$ is the lowest degree used in the fit, chosen to achieve the best fit possible. In general power law exponents of around 2-3 are considered good for scale-free properties. This is the case for out-degree and total degree of both graph. However, note that the corresponding $k_{\min}$ differ significantly here. For the out-degree the $k_{\min}$-values are fairly low but for the total degree of USPTO, $k_{\min} = 11$ was needed to achieve this fit. Also the exponents for the in-degree distribution fits are higher, thus requiring further evaluation. Figure 6.2 shows the Complement Cumulative Distribution Function (CCDF) for total degree distributions of ELN and USPTO compared to the theoretical power laws of the estimated parameters. We see that they mostly fit well but that the empirical curve dips down in the right tail for USPTO.

	ELN		USPTO	
	γ	$k_{\min}$	γ	$k_{\min}$
Total	2.33	5	2.16	11
In	8.70	4	6.97	2
Out	2.22	3	2.11	5

Table 6.1: Estimated parameters for fitting the degree distributions to power law, presenting the obtained power law exponents, γ, as well as the optimal minimum degree used, $k_{\min}$.

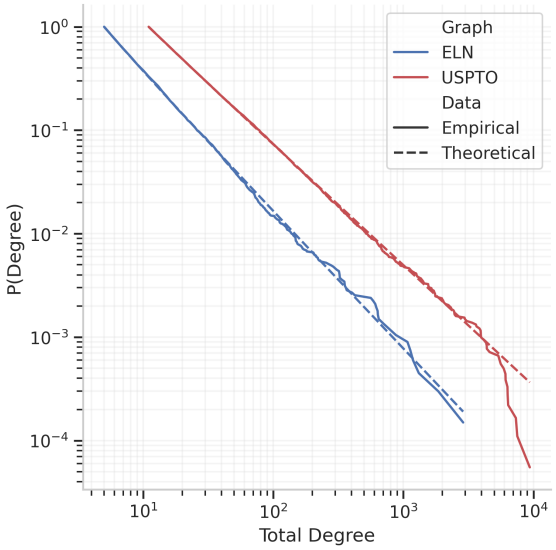


Figure 6.2: CCDF for the empirical total degree distribution of the two graphs as solid lines, compared to the fitted theoretical power law distributions (dashed).

Likelihood-ratio tests are shown in Table 6.2, for comparing how well the degree distributions fit to power law compared to other distributions. Positive R-values

mean that power law is favored compared to the distribution in the column and the significance of the result is indicated by the p-value, which should be approximately below 0.1 to be considered significant. Based on these results we can rule out that the degree distributions are best described by exponential, stretched exponential and positive log-normal, however with some uncertainty regarding ELN's in-degree distribution.

	Log-normal		Positive Log-normal		Truncated Power Law		Exponential		Stretched Exponential	
	R	p	R	p	R	p	R	p	R	p
ELN										
Total	-0.34	0.65	120.47	7.50e-10	-0.33	0.42	3322.68	7.31e-20	23.53	0.01
In	-0.15	0.69	-0.15	0.69	-0.15	0.59	-0.15	0.6	-0.16	0.68
Out	-5.81	0.07	241.18	1.07e-15	-2.81	0.02	6098.31	5.94e-27	30.24	0.02
USPTO										
Total	-0.40	0.61	254.45	4.69e-22	-2.13	0.04	12056.12	2.95e-74	90.40	3.68e-09
In	-119.18	5.08e-19	4699.08	2.25e-154	1.03e-04	0.99	8781.57	1.07e-45	60181.86	5.51e-32
Out	-4.27	0.08	845.09	1.01e-62	-5.80	6.6e-04	31030.38	2.52e-114	142.53	2.10e-11

Table 6.2: Likelihood-ratio test results from comparing the power law fit to fits to other distributions. A positive R indicates that power law is a better fit for the data and a negative R indicates that the other distribution describes the data more accurately. The p-value is the significance of the R value. If $p > 0.1$ the direction of R is not reliable.

Furthermore, when looking at the results in Table 6.2 it is unclear if any of the other distributions were better fits than power law for the in-degree distributions as most of the p-values for all fits are high and consequently indicates uncertainty. For the out-degrees both log-normal and truncated power law are slightly better fits. If the total degree of ELN is considered instead, there is some ambiguity whether log-normal and truncated power law are better than power law or not. Based on this result, the total degree distribution of ELN could approximately be described by power law, even though the fit is not perfect and potentially worse than log-normal and truncated power law. For USPTO the results are similar except for the in-degree distribution, where the log-normal fit is clearly preferred over power law for total degree. However, we cannot with any significance distinguish if log-normal is a better fit than power law. Truncated power law seems to be slightly preferred over power law, except for the in-degree where the results are unclear.

Since these results do not give as clear indications that the degree distributions are best described by a power law, as were the case in similar studies [6], the same procedure was done on the molecule graph to rule out the possibility that the deviation from power law could be described by the bipartite architecture. These results can be found in Table A.1. They are highly similar in character and thus the deviations from power law cannot be explained simply by the bipartite graph structure.

To conclude this section, the total degree distributions of both ELN and USPTO can approximately be described by power law, however it is possible that log-normal or truncated power law would fit the the degree distributions as well or even better.

Most importantly, the degree distributions are not described by the exponential or stretched exponential distribution, which means that they are certainly heavy-tailed.

6.2 Hierarchical Structure

As described in Section 5.2.1 the hierarchy of the grahs can be explores in a number of different ways. First, the small-world property was first evaluated in terms of average shortest path length and number of hubs. After this, betweenness centrality and clustering coefficients were computed and analyzed in terms of their dependence on vertex degree.

6.2.1 Small-world Behavior

As previously seen in Figure 6.1, both graphs exhibit a proportion of highly connected vertices. Considering the total degree, a threshold was chosen above which vertices were classified as hubs. This threshold was taken according to $\delta = 100 d_{\text{avg}}$, where d_{avg} is the average total degree for the respective graphs, namely 2.1 and 2.4 for ELN and USPTO respectively. With this threshold ELN had 43 hubs making up 0.01% of the graph and USPTO had 447 hubs making up 0.02% of the graph.

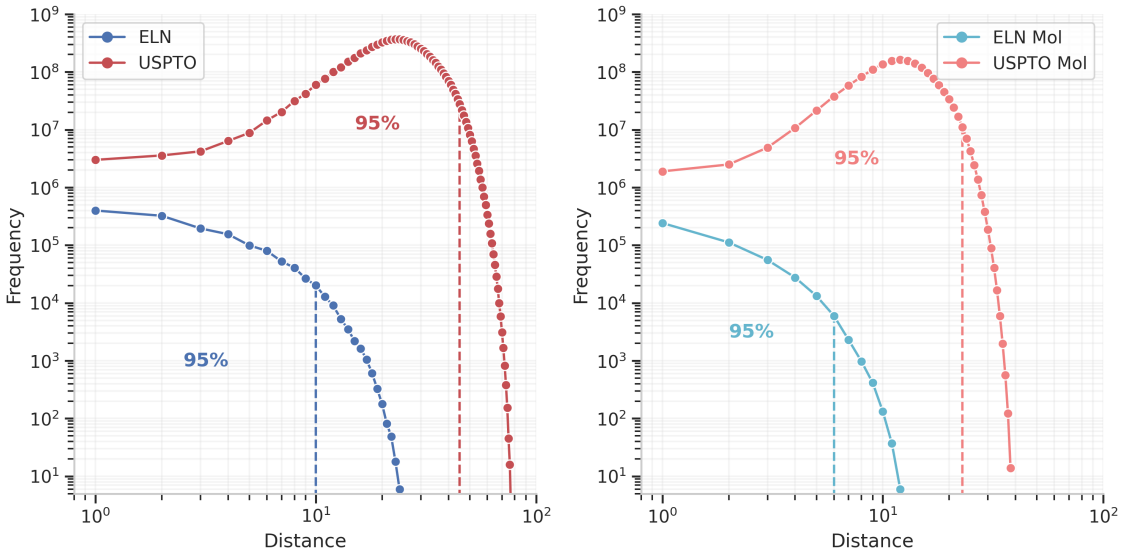


Figure 6.3: Distance histograms for the two graphs, when all vertices are considered in left panel and for the molecule graphs in the right panel. The vertical, dashed lines show the point beyond which 95% of all shortest path have been counted.

Consequently, since the graphs have hubs, there are reasons to think that the graphs could also exhibit small-world properties, for which the shortest path length should be small. Figure 6.3 shows the distributions of the shortest paths in the graph, 95% of all paths are shorter than the distances marked by vertical, dashed lines in the plots. In order to investigate how the bipartite graph architecture affects the path

lengths, the results for the molecule graphs were also considered. As can be seen in Table 6.3, the average shortest path lengths are very short for the ELN graph version, indicating that it is has the small-world property. The bipartite graph structure is likely the reason why the path length is about twice as long for this graph compared to the molecule graph.

	ELN	USPTO<1998	USPTO
Bipartite	3.38	10.88	25.05
Molecule	1.90	5.50	12.81

Table 6.3: Average shortest path lengths for the two graphs and USPTO<1998, compared between the bipartite graph structure and the molecule graphs.

The same calculations on USPTO show significantly longer path lengths than what would be expected from a small-world graph and compared to ELN. To investigate if this observation could be explained by the size difference alone, the USPTO subset containing reactions registered before 1998 was used. This subset is approximately five times smaller than the original graph and roughly the same size as ELN. The average shortest path length in this subset was shorter but still longer than in ELN. Thus, the much longer average path length in USPTO cannot solely be explained by the number of nodes alone. Instead USPTO is believed to contain some longer, isolated reaction pathways. Further investigation of the underlying reason for these unexpectedly long reaction pathways is beyond the scope of this project.

6.2.2 Measurements of Hierarchy

In addition to the presence of hubs and small-world property, the hierarchy was measured in terms of average degree dependent betweenness and clustering. First, the result from calculating and plotting the average degree dependent betweenness, can be seen in Figure 6.4. One can observe that the betweenness centrality increases with the vertex degree. This is an indication of hierarchy in the graph since it means that higher connected vertices have more paths running through them. The trend is the same for both ELN and USPTO and is also observed in the molecule graphs.

As expected, the clustering coefficients for the bipartite graphs were all exactly zero, since by design no triangles can exist in these graphs. Instead the clustering coefficients were evaluated on the molecule graphs. The global clustering coefficients were calculated to be 0.00018 for ELN and 0.00010 for USPTO, notably lower than the typical range 0.01-0.5 for many other graphs. The local clustering coefficients were also calculated and their average degree dependence is shown in Figure 6.5. This figure shows that the average local clustering coefficient decreases with the vertex degree, meaning that the vertices with high degree are not part of clusters to the same extent as ones with fewer connections. As previously stated some real-world graphs have shown that the averaged degree dependent clustering coefficients scales according to $\bar{c}(k) \propto k^{-1}$ for hierarchical graphs [6]. It agrees with what can be observed in Figure 6.5, where the slopes of the linear regression lines were found to be close to -1 for both ELN and USPTO.

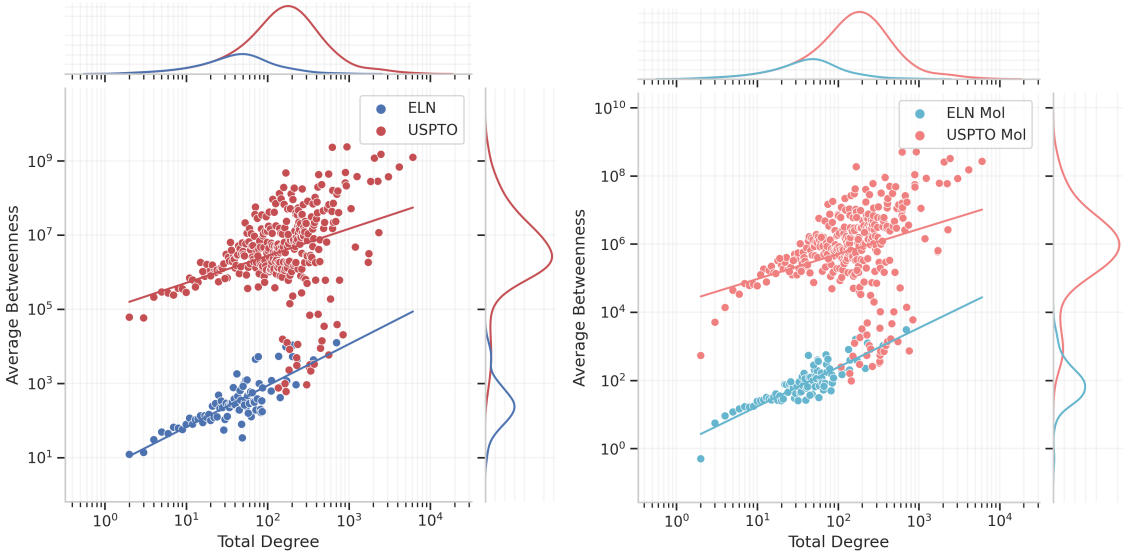


Figure 6.4: Average degree dependent betweenness. The margins show histograms of the data and the lines represent a linear regression showing an upward trend.

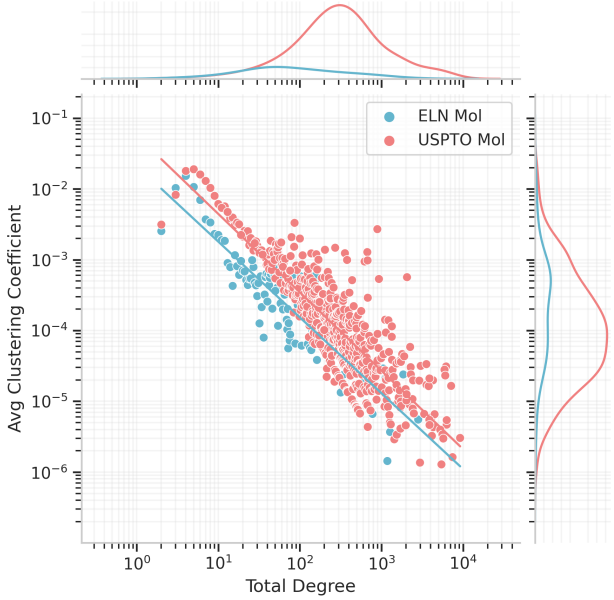


Figure 6.5: Average degree dependent clustering. Linear regressions of the respective data are shown as line, where the ELN line has slope -1.07 and the USPTO line has slope -1.11.

6.3 Network Components

In order to identify the different components of interest, the first step was to conclude whether the respective graphs had a core. To do so, the left panel in Figure 6.6 shows a histogram of the sizes of SCCs in each graph. For USPTO one such component

clearly fits the definition of a core, its size is 10^2 magnitudes larger than remaining ones. This has been emphasized in the figure by an additional larger marker. ELN on the other hand does not have any such SCC. The largest SCCs are of size 12 and there are two of them.

Both graphs do however have a CC significantly larger than any other, these can be referred to as the *central components* of the respective graphs, as seen in the right panel of Figure 6.6. In the USPTO case the core was confirmed to be contained in this component and therefore it's periphery is defined as the relative complement of this central component and the core. Even tough ELN did not have a clear core this central component for the graph can still be taken to have a more central meaning than the remainder of the graph and thus it is defined as the graph's periphery. For both graphs, all remaining vertices are considered belonging to islands and the resulting proportions of each structure are presented in Table 6.4. Note that the core in USPTO only make up 0.2% of the graph.

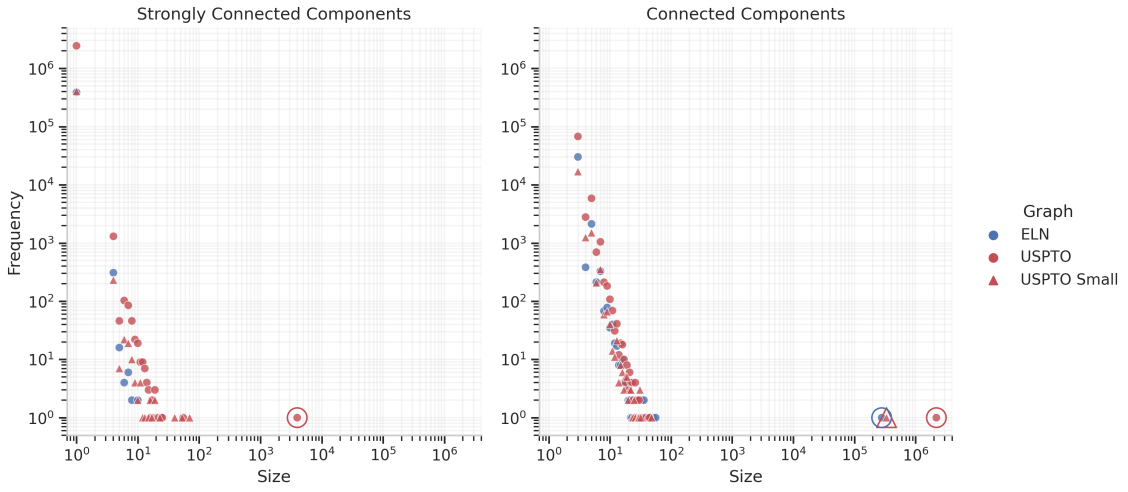


Figure 6.6: Histograms over component sizes, connected ones in an undirected sense in the left panel and SCCs in the right panel. The results for the subset of USPTO <1998 is also shown.

	ELN 389,123	USPTO < 1998 405,297	USPTO 2,457,226	Beilstein 5,900,000
Core	0.0%	0.0%	0.2%	4%
Periphery	71.8%	82.8%	89.1%	78%
Islands	28.2%	17.2%	10.7%	18%

Table 6.4: Proportions of core, periphery and island vertices in the ELN and USPTO graphs, as well as the smaller subset of USPTO. Note that no core components were found in either of the ELN or the USPTO < 1998 graphs. The result for the Beilstein database from [5] is also shown for comparison.

An initial hypothesis regarding the absence of a core in ELN was that it could be due to the small size of the graph. To test this hypothesis the same structure analysis was done on the prepared smaller subset of USPTO, with a size relative to ELN's also seen in the above results. From this result it is evident that USPTO <1998 also lack a core, which corroborates the hypothesis that the overall size of the graph affects this property.

6.3.1 Molecular Descriptors

We next calculated metrics related to molecular complexity, seen in Table 6.5 averaged over each component as well as over all molecular hubs found in the respective graphs. The hypothesis from literature, [22], is that hub and core molecules should be less complex and instead building-block substances, i.e. smaller and diverse. Islands on the other hand are presumed to be made up of more complex, final-drug compounds. Drug-likeliness related metrics were also calculated in a similar way but since this did not result in any interesting conclusions these results can be found in Table A.3.

MW is one of the most simple indicators of molecular complexity. This is clearly highest for molecules in islands, for both ELN and USPTO, and becomes lower as you pass through the periphery into the core, with the lightest molecules on average being the hubs. Following the same trend is the number of heavy atoms, number of rings, and number of chiral centers, further corroborating the hypothesis. The trend is the same for the network components regarding the Fsp3 value but deviates notably for hub molecules, which instead have the highest Fsp3 values in both graphs. This stands out at first glance since higher such values are typically found for more complex molecules. However, it is known to be less so for very small molecules where each carbon hybridization weighs heavier and as previously noted the hubs are clearly significantly smaller by the look of the molecular weight.

			Complexity			
	# Molecules	MW	#HeavyAtoms	#Rings	#ChiralAtoms	Fsp3
ELN						
Full	233,840	367.68±3.2e-01	25.5±2.3e-02	2.96±3.2e-03	0.68±2.3e-03	0.38±4.5e-04
Islands	72,279	401.02±6.2e-01	28.08±4.4e-02	3.19±5.8e-03	0.85±4.7e-03	0.39±7.4e-04
Periphery	161,561	352.77±3.6e-01	24.35±2.6e-02	2.85±3.8e-03	0.6±2.5e-03	0.38±5.6e-04
Hubs	47	107.0±8.3e+00	6.45±5.6e-01	0.55±1.2e-01	0.04±4.2e-02	0.65±6.0e-02
USPTO						
Full	1,372,550	359.82±1.1e-01	24.87±8.3e-03	2.88±1.3e-03	0.61±9.9e-04	0.35±1.9e-04
Islands	173,777	389.89±3.5e-01	27.27±2.5e-02	3.07±3.7e-03	0.94±3.8e-03	0.37±5.2e-04
Periphery	1,197,495	355.61±1.2e-01	24.54±8.7e-03	2.85±1.3e-03	0.57±9.8e-04	0.35±2.0e-04
Core	1,278	216.34±2.2e+00	15.1±1.5e-01	1.55±2.7e-02	0.34±2.2e-02	0.31±8.1e-03
Hubs	523	129.56±2.6e+00	7.74±1.6e-01	0.6±3.0e-02	0.02±6.6e-03	0.48±1.8e-02

Table 6.5: Molecular descriptors related to their complexity, averaged over all molecules in different sub-components of the two graphs. MW refers to the average molecular weight.

7

Link Prediction Outcome

For the next phase of the thesis, link prediction was performed in the undirected molecule template sub-graph of ELN, which for radius 1 has 6497 vertices and 17274 edges while for radius 2 has 19553 edges and 39009 edges. Thus, both graphs are sparse with densities of 0.00082 for radius 1 and 0.0002 for radius 2.

The results have been divided between the development and optimization of the model and the evaluation of the final version. The latter includes a comparison with baseline models, interpretation of chemical feasibility and finally a hand full examples of novel reaction templates predicted by the final model as additions to the graph. As in the previous chapter, results are alternated with their respective discussions.

7.1 Training Configurations

As a first step, after the algorithm and model were set-up and adapted to the given data and all of the settings and hyperparameters were explored. The following section goes through the steps; optimizing model hyperparameters and modifications, comparing the negative sampling methods as well as oversampling of the negative class, and combining trained models to create a voting ensemble model.

7.1.1 Hyperparameter Search

First, hyperparameters involved in the GNN, as well as modifications done to the network, were optimized through a cross-validation on the ten random fold as explained in Section 5.3.1. This was done only for models trained on randomly sampled negatives, without oversampling and only on the graph version with molecular templates of radius 1. The optimization was done from a standard setting, namely 8 hidden channels, 2-hop sub-graphs, 50% dropout between one-dimensional convolutions, and with fingerprint attributes on vertices. Each of the settings were then changed one at a time according to Table 5.2. Figure 7.1 shows a summary of the training and validation results over the cross-validation for each scenario tested. The standard case has been added to each panel, in gray, for easier comparison. For each case, the curve shows the average over all ten runs with standard deviation as surrounding error bands. All of the individual runs are available in Figures C.2 and C.3.

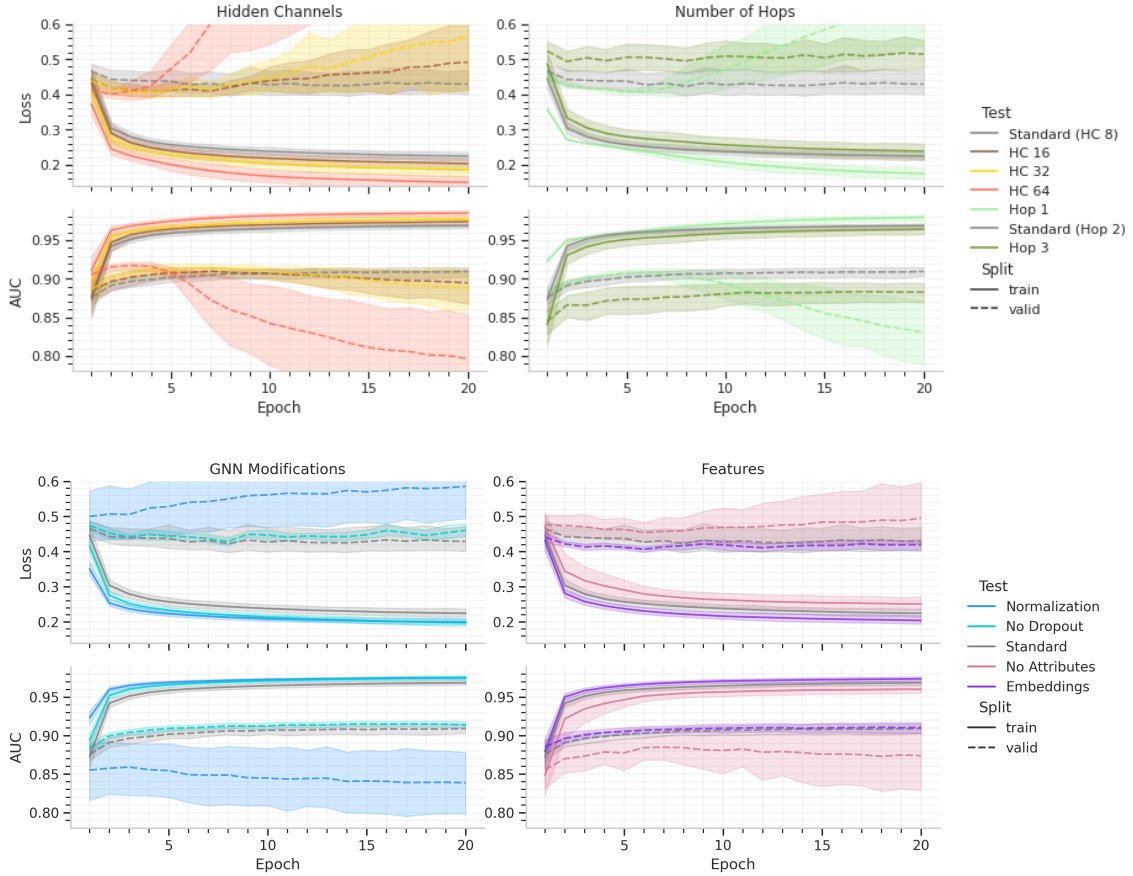


Figure 7.1: Training progression of models with randomly sampled negatives, comparing different scenarios of hyperparameter settings.

As expected, the loss and AUC-values follow similar trends for all cases. There is some bias in all models, seen as the gap between training and validation performance. All cases have converged or passed their optimum after the 20 epochs shown. Not surprisingly, increasing the number of hidden channels improves the training performance whilst causing overfitting, as seen in the validation curves of the top left panel. For the number of hops, an inverse trend can be seen. With sub-graphs of 1-hop overfitting can be seen, but with 3-hops the validation performance suffers instead, never reaching as good values as the other options. Regarding the contributions proposed in the present work, namely normalization and dropout, it can be seen in the bottom left panel that normalization in general hurts the learning performance of models. Thus, the hypothesis that this would help to combat overfitting was not the case for this model and dropout did not materialize for this data in the given settings either.

When it comes to vertex features added to the sub-graphs, the bottom right panel of the figure, clearly shows that training on fingerprints helps the learning performance. Since the fingerprints are the only source of chemical information given to the model it is also intuitive that this were a meaningful addition for reaction prediction, contrary to otherwise only having structural information about vertices and their

relationships. Embeddings on the other hand, only generated slight improvements compared to the standard settings with no embeddings. A possible explanation for this is that enough structural and graph topological information is included in the vertex labeling done by the SEAL algorithm, i.e. the DRNL. Recall that the authors of the SEAL algorithm even suggested that the addition of embeddings could hurt the performance, which was at least not the case here.

The best model from a run was chosen based on the epoch of highest validation AUC, as described in Section 5.3.2. For this reason the best validation AUC reached for each case of hyperparameters are also presented in Table 7.1. In the table both the result averaged over all runs and the maximum value achieved from all runs are included. The best settings in terms of highest maximum validation AUC achieved was using normalization. However, since the poor average result for this case indicates instability, we decided against using this option. From the remaining cases, the highest maximum AUC values were achieved for the cases with many hidden channels. With 64 hidden channels the absolute highest values were obtained, however the result with 32 hidden channels was close second. Since an increase in hidden channels increases the number of trainable parameters, which in turn prolongs the training times, 32 was deemed sufficient. This was also the case in the original implementation of SEAL, that 32 hidden channels were used.

From this reasoning the hyperparameter settings were chosen to be 32 hidden channels, but otherwise with the standard settings. Note that further testing of more combinations is likely to achieve even better results, but for the scope of this thesis these configurations were satisfactory.

Configurations		Validation AUC	
		Avg±Std	Max
Standard		91.01±0.0095	92.22
Hidden Channels	16	91.43±0.0074	92.82
	32	91.85±0.0086	93.04
	64	**92.23**±0.0056	93.17
Number of Hops	1	91.47±0.0078	92.94
	3	88.57±0.0209	91.05
GNN Modifications	No Dropout	91.65±0.0058	92.53
	Normalization	87.07±0.0532	**94.17**
Vertex Features	No Attributes	89.13±0.0252	92.01
	Embeddings	91.27±0.0088	92.58

Table 7.1: Validation AUC from cross-validations on ten random folds for different hyperparameter settings. All models were trained on randomly sampled negatives, without oversampling. Top results have been highlighted in bold.

7.1.2 Molecule Template Radius

With the chosen settings from above, a similar cross-validation was performed between the ELN graph of molecular templates with radius 1 versus radius 2. Also in

this case, both versions were trained on randomly sampled negatives, without oversampling. Table 7.2 presents the best validation AUC achieved in the two scenarios. Note that the case for radius 1 is a duplication of the case hidden channels 32 in Table 7.1. The full set of learning curves can be found in Appendix C. According to these results, training on an increased radius of molecular template does not improve the model performance.

		Validation AUC	
		Avg±Std	Max
Radius	1	**91.85**±0.0086	**93.04**
	2	89.61±0.0092	90.93

Table 7.2: Validation AUC from cross-validations on ten random folds for molecule templates of radius 1 versus 2. All models were trained on randomly sampled negatives, without oversampling. Top results have been highlighted in bold.

Next, the learning process's dependence on the size of the training set was explored by training the models on data from an increasing reaction date window. This type of test was meant to evaluate how much of the data was needed in order for the model to learn it properly. As such the best validation AUC, achieved from training on each of the time folds presented in Table 5.1, are shown in Figure 7.2. Note that the size of the graph with radius 2 is much larger since the vertices there correspond to more specific molecules. It can be seen that results are consistently lower for the radius 2 case, even for training data sets of similar size to the maximum size of the radius 1 set. Since the hyperparameters were only optimized for radius 1, it cannot be ruled out that models trained on radius 2 could achieve better results with further optimization.

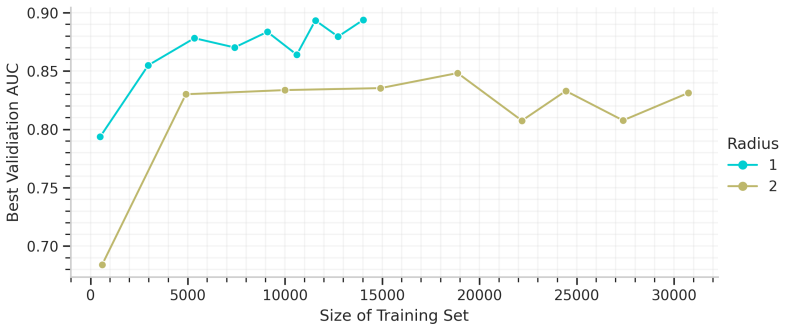


Figure 7.2: Validation performance depending on training set size in terms of date of entry. Showing the best validation AUC achieved during training on the respective time folds, for molecule templates of radius 1 and 2 respectively. All models were trained on randomly sampled negatives, without oversampling.

It is expected to see worsened performance for a small training set, as seen for both radius cases in Figure 7.2. For radius 2, the performance roughly reaches its maximum already for the next smallest set. Also for radius 1 the results for all

but the smallest training set are somewhat similar. These findings indicate that it is likely sufficient to train on significantly smaller proportions of a graph in this context. This is a useful property given that graphs grow as more knowledge is added. However, for the remainder of this thesis we will continue to train on the molecule templates of radius 1 from the entire graph since it is still computationally feasible and as it results in the best result.

7.1.3 Analysis of Negative Sampling Strategies

The initial hypothesis was that negative links sampled at random would deviate substantially from the positive links and that it therefore would be beneficial to sample the negative class from the same distribution as the positive class, and so end up with a harder classification problem. As described in Section 5.3.1, two additional negative sampling strategies were proposed as a part of this thesis; negative sampling from distribution and structured negative sampling. Before training on these methods, a brief analysis on the links sampled from each method was conducted in order to test the initial hypothesis.

The following results are based on all links in the graph (17274) and for each of the sampling strategies five times as many negative links were sampled (86370). An initial concern was that there could be duplicates within the sampled links, especially for the distributed negatives, this was tested and no duplicates were found within any of the set of sampled negative links. Figure 7.3 shows distributions of two different aspects of the negative sampling strategies.

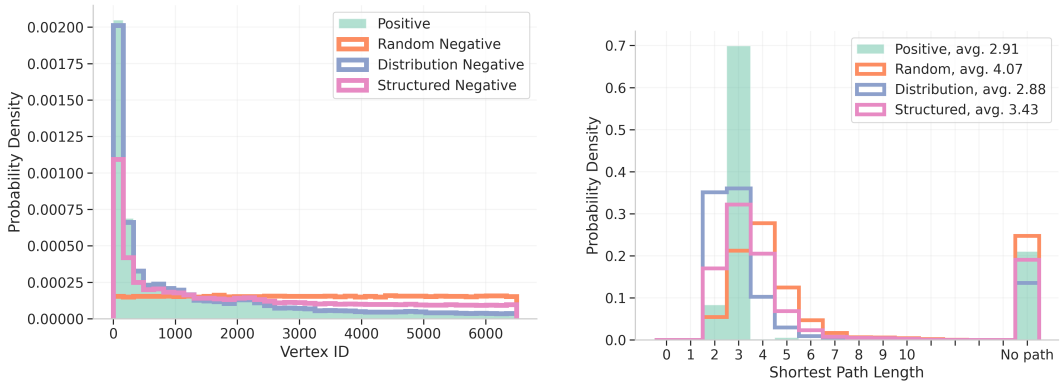


Figure 7.3: Left: Probability density of each vertex occurrence in a link, either as source or target. Right: Distribution of the shortest path length between source and target vertices for the sampled links.

In the left panel of Figure 7.3 the distribution of the sampled links are compared to the positive class. As expected, sampling from the distribution results in a negative class that is more similar to the positive class in terms of the frequency of vertices involved in the links. We also investigated the relationship of the source and target vertices, in terms of their distance, and therefore the average shortest path lengths

were computed. The right plot in Figure 7.3 shows the probability density for the shortest path length between the source and target vertices in the sampled negative class, compared to the positive class. For each positive link, the link was first removed and then the shortest path length was calculated between the two vertices. This result also indicates that sampling from distribution gives a negative class that is more similar to the positive class, in terms of the average distance between the source and target vertices. Random negative sampling generates source and target vertices that, on average, are further apart while structured negative sampling is somewhere in between. This would confirm our hypothesis in Section 4.3 that sampling from random is more likely to, when extracting sub-graphs for SEAL, result in sub-graphs which are disconnected as this becomes more probable with increasing distance.

Next, we plotted the source degree against the target degree as to investigate the relationship between the source and target vertices, Figure 7.4. This result indicates that sampling from distribution gives a negative class more similar to the positive class and that sampling from random gives a negative class considerably different from the positive class, as the degrees of source and target are on average substantially lower than for the other distributions.

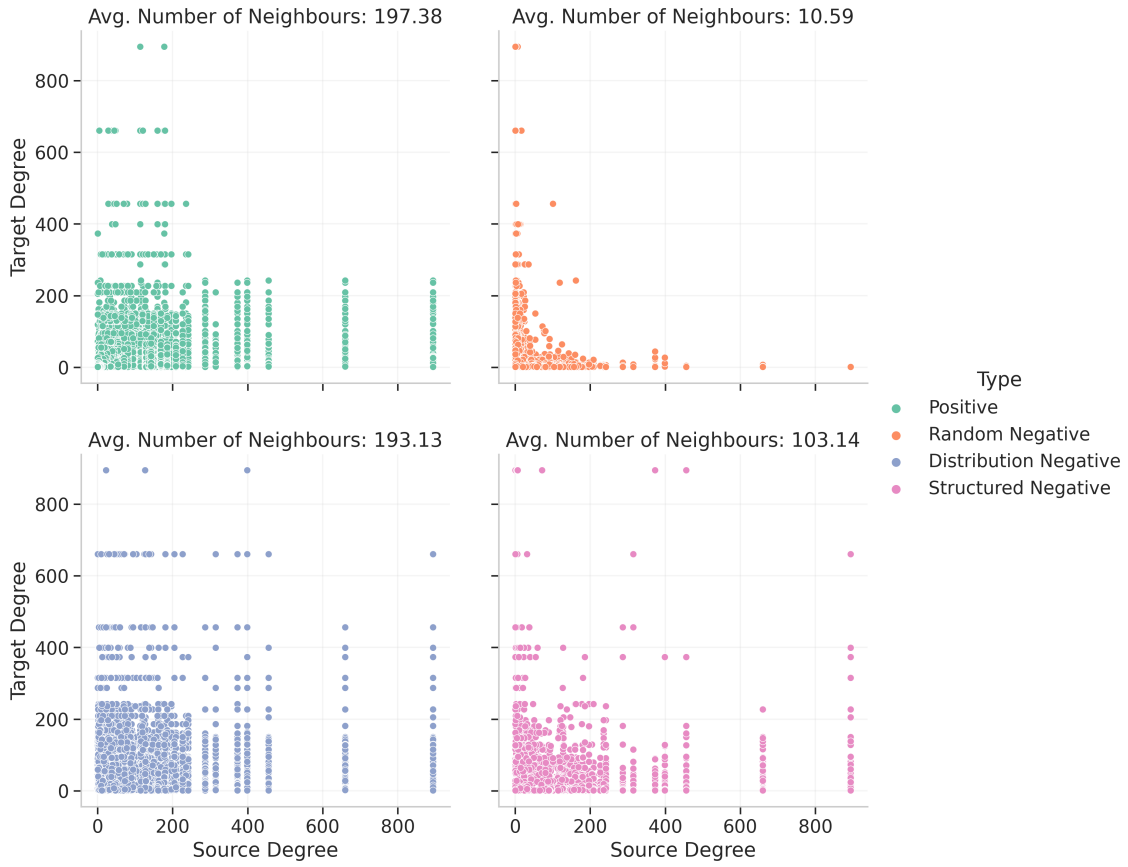


Figure 7.4: Source versus target degree for each of the negative sampling methods. The average number of neighbors for source and target vertices together is included in the titles.

As discussed in Section 5.3.4, it is probable that some of the sampled negative links are actually positive links. To estimate how the propensity to sample such missing links vary with the negative sampling method, all sampled negative links were searched for in USPTO. It was observed that there is a higher probability to sample missing links when the negative sampling is done from the link distribution than the other methods, Table 7.3. For comparison, the number of positive links in ELN that are also links in USPTO is roughly 50 %. It is expected that these numbers would be higher if it would have been possible to search through all known chemically reactions.

		Negative		
	Positive	Random	Distribution	Structured
% in USPTO	50.52	0.19	6.00	2.02

Table 7.3: Percentage of links in the respective link classes that are present in USPTO.

To conclude this analysis of the negative sampling methods, all results supports the initial hypothesis that negative sampling from distribution results in a negative class more similar to the positive class. There are still some differences such as that the distributed negative class can have both source and target vertices with high degree contrary to the positive class. It also has somewhat different average shortest path length and the average number of neighbors for the source and target vertices varies slightly between positive and distribution. Sampling the negative class at random results in a negative class that differs from the positive class in all investigated aspects and sampling using the structured method results in a negative class the result is found between what is obtained by random sampling and sampling from distribution. The results further suggest that sampling from distribution increases the chance to sample links that represent reactions known outside of ELN, meaning that it does not only consist of unfeasible reactions.

7.1.4 Training on Different Negatives

As there are some significant differences between the sampling methods, we expected the choice of strategy to affect the training outcome and the link prediction. If oversampling of the negative class has a positive impact on the training will also be evaluated here, by sampling five times as many negative links as positives. For these tests, hyperparameters and settings were chosen according to the best option found by the previous sections, it is also done only on the molecular templates of radius 1 and with the full dataset available. As in the hyperparameter search a cross-validation was first performed on the ten random folds, for which the learning curves are shown in Figure 7.5 and the best validation AUC values achieved by each scenario are presented in Table 7.4. The full set of learning curves for all cases can be found in Figure C.4.

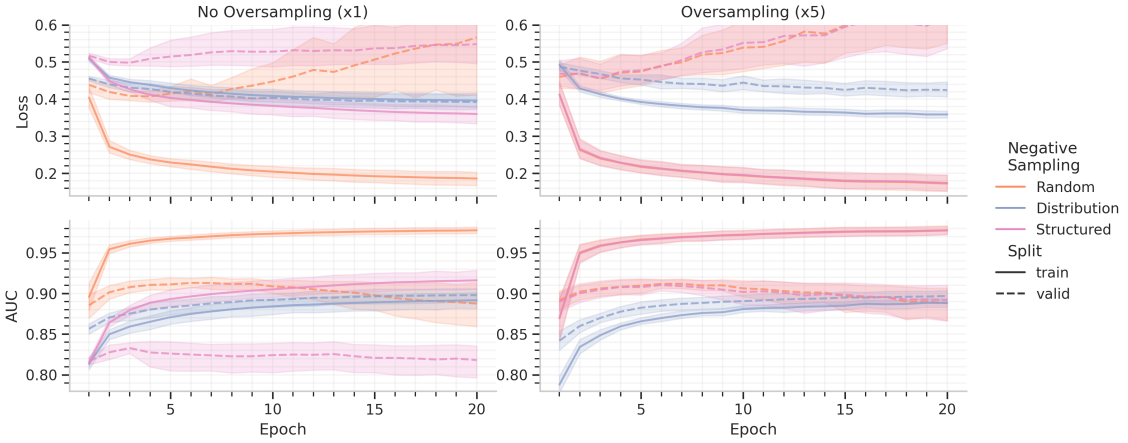


Figure 7.5: Learning curves of models trained on negatives sampled according to the three strategies, comparing simple sampling in the left panel with five times oversampling in the right panel.

Without oversampling, the model trained on randomly sampled negatives can clearly be seen to outperform the other two models, at least in terms of validation AUC. The loss, on the other hand, is comparable between the model trained on random and the one trained on distribution negatives. These two models do not seem to be significantly improved by oversampling. When it comes to oversampling of the structured negatives a substantial improvement can be seen. Surprisingly, this model becomes almost identical in performance to the one trained on randomly sampled negatives. An explanation for this might be that oversampling of the structured negatives introduces more stochasticity, making it more similar to random sampling.

Another interesting takeaway from this result is the very low bias seen in the models trained on distribution negatives, according to the very small difference in training and validation performance. In some aspects, the validation performance for these models even surpass the training performance. This in itself, should not be alarming since it is likely due to the dropout which, per default, was only applied during training (not validation) and thus is expected to hold back the training performance. The low bias should be seen as a very promising result for this sampling strategy, even though the specific models trained here did not measure up to the other two methods. The reason for this is that the hyperparameters and settings again were only optimized for the random case, and thus further exploration of this case individually could potentially give better results.

The validation sets used for the results above were all made up of negatives sampled in the respective ways as the training sets. To further evaluate the models they were tested on three different test sets. All of these contained the same positive links and the same number of negatives were sampled in the three different ways. Thus, all models were tested on negatives sampled in all three ways. Table 7.4 also presents this result in terms of AUC for all models.

Trained on	Sampling Factor	Validation AUC		Test AUC		
		Avg±Std	Max	Random	Distribution	Structured
Random Negatives	x1	**91.85**±0.0086	93.04	86.65	69.31	72.36
	x5	91.81±0.0100	**93.25**	**87.86**	68.40	71.60
Distribution Negatives	x1	89.84±0.0144	91.64	47.60	**87.91**	69.42
	x5	89.71±0.0127	91.16	48.20	87.08	71.03
Structured Negatives	x1	84.34±0.0076	85.98	74.41	77.61	**76.06**
	x5	91.71±0.0100	93.15	87.62	69.15	71.74

Table 7.4: Resulting maximum and average validation AUC from cross-validations on ten random folds for models trained on the different negative sampling methods. "x1" refers to no oversampling and "x5" refers to five times oversampling of the negative class. Also, the test AUC of the models on the respective test sets containing negatives sampled in the respective ways is presented. Best results have been highlighted in bold.

Firstly, it can be concluded that the model trained on the given sampling strategy achieves the best result when evaluated on that test set. Surprisingly, the highest AUC overall is actually the model trained on distribution negatives tested on the distribution set. This means that it is in fact possible for this model architecture to learn even these presumed more difficult links well. On the other hand, this model performed much worse on the other two test sets. The hypothesis was that this model would, not only learn the hard negatives, but to also be able to distinguish the supposed more easy random links, which it did not. Equally surprising is the fact that oversampling of the structure negative class only improved the performance on the random test set. This gives some further support to suggestion that the oversampling of this strategy makes the class more similar to the class of randomly samples negatives.

When taking all of the results into consideration the oversampling does not appear to have affected the distribution trained model. For the random and structured trained ones however, we conclude that the oversampling seems to be a reasonable addition. Thus, going forward these version will mainly be used. So far, the results regarding the different negative sampling strategies are deemed inconclusive, and they will be explored further in the remainder of this chapter.

7.1.5 Voting Ensemble Model

As the models trained on the different negative sets appears to have learned to identify the negative class based on different aspects, our hypothesis is that it could be beneficial to combine the predictions from each model to an *ensemble* model. This model then makes predictions based on an average from the other models' predictions.

The models were combined in three different ways; the three models trained without oversampling, the three models trained with oversampling, and the best model from each sampling method. In the latter case the best model was chosen according to the

overall assessment of Table 7.4 described in the previous subsection. This includes the distribution model without oversampling and the other ones with oversampling. Results for these voting models can be found in Table 7.5 and will now be compared to the previous results from Table 7.4. Looking at Voting Ensemble x5 specifically, the AUC on the random set is comparable with the Random x5 model's AUC on the same test set, 87.86. However, the Voting model has a notably higher AUC-value when evaluated on the distribution set. Recall that the random model only achieved 68.40 on this test set. The same can be observed for Voting Ensemble Best. Figure 7.6 further compares the results of the Random x5 and Voting x5 models.

		Test AUC		
		Random Set	Distribution Set	Structured Set
Voting Ensemble	x1	79.86	**82.43**	**78.09**
	x5	**87.15**	79.23	74.68
	Best	86.70	81.00	74.57

Table 7.5: Test AUC of voting ensemble models. Best refers to when oversampling was used only for the random and structured trained models. The best voting model on each test set has been highlighted in bold.

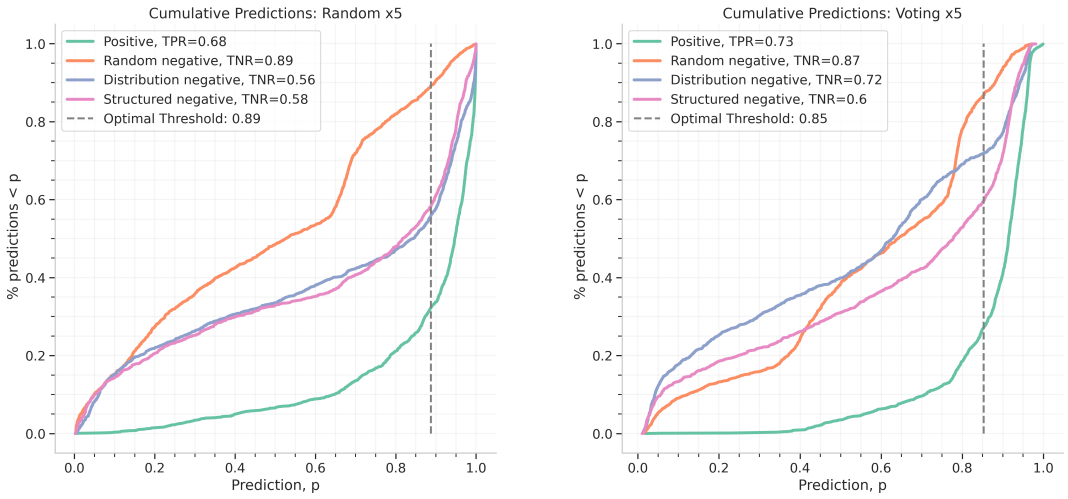


Figure 7.6: Test predictions by the Random x5 model (left) and the Voting x5 model (right). These plots shows the percentage of predictions below prediction, p, against p. The cumulative curve of the predictions for the positive class is compared to those of the different negative classes.

Comparing the two plots in Figure 7.6 further confirms the previous conclusions. The plots show the percentage of predictions which are predicted below p against p for the positive class as well as for each negative class and can be used as a visual example of how good the models are at recognizing each of the negative classes.

Ideally, the line for the positive class should be as close to the bottom right corner as possible while the line for the negative classes should be as close to the top left corner, since this would indicate a perfect separation between the classes. For the Random x5 model (left plot) one can see the random negative class is clearly better separated from the positive class than the distributed and structured classes.

When looking at the Voting x5 model instead (right plot) there is not as clear separation between the different negative classes as these are more intertwined. However at the marked threshold, the voting model is still better in distinguishing the random negative class from the positive. When looking at the positive class alone the voting model is more confident in recognizing the positive class. This can be seen by looking at the TPR, which for the Random x5 model is 0.68 and 0.73 for the Voting x5 model. The same thing can be seen by comparing the green line in both plots, as it is closer to the bottom right corner in the right plot than the left. Overall, the voting model has increased its ability to separate the different negative classes from the positive one compared to the Random x5 model.

7.2 Evaluation of the Final Models

Results from evaluating the best models from the previous optimization and search are presented in the current section. This evaluation focused on the test set with a randomly sampled negative class and the positive class since this is the more common way to evaluate link prediction. First, a comparison between the models and a number of heuristic baseline alternatives is shown. Next, the optimal thresholds for each models predictions are determined with which the chemical feasibility of predictions are explored. Ultimately, a few examples are given of novel reactions predicted by the resulting model.

7.2.1 Comparison with Baseline

We first compare performance of our trained SEAL models with several common heuristics-based baselines for link prediction, shown in Table 7.6. The comparison was done for several different metrics. The test set on which the metrics were computed contained only randomly sampled negatives, equally many as the positives.

As described in Section 5.3.3, AUC and AP are good evaluators of binary classifiers and in this context link-equality focused. For these two metrics there is a distinct separation between some of the SEAL models and the baselines. SEAL models trained on random and structured negatives as well as the voting ensemble achieve values all in the high eighties whereas the baselines perform similar to a random classifier.

The Hits@K scores are also link-equity metrics, but even more adapted to the link prediction task since they implicitly incorporate the uncertainty in the negative class. The values of these metrics are less interpretable on their own and more useful for the type of comparison made here, between different models. They are naturally lower for lower K-values since the task becomes harder with decreasing

K. It can be seen in Table 7.6 that on these metrics all SEAL models, apart from Distribution, by far outperforms the baselines.

		AUC	AP	MAP	Hits@20	Hits@50	Hits@100
Heuristic Baselines	CN	46.97	49.06	49.56	0.0078	0.0136	0.0279
	AA	47.06	49.40	49.68	0.0091	0.0188	0.0370
	JI	46.68	48.05	49.48	0.0006	0.0032	0.0065
SEAL Models	Random (x5)	**87.86**	**88.53**	89.94	**0.3132**	0.4540	**0.5979**
	Distribution	47.60	46.13	67.88	0.0149	0.0201	0.0298
	Structured (x5)	87.62	88.36	**89.95**	0.3022	**0.4656**	0.5921
	Voting (x5)	87.15	87.41	89.39	0.2341	0.4501	0.5934

Table 7.6: Further test results of the best SEALmodels compared to three baseline models described in Section 4.1. Metrics are based only on results on the test set with randomly sampled negatives. The best model in terms of each metric has been highlighted in bold.

Especially interesting are the Hits@K scores for the AA baseline. This method can be seen to have outperformed other link prediction methods, including SEAL, on one of the OGB data sets "ogbl-collab" in the official leader-board with a Hits@50 score of 0.6417 [59]. On this dataset the same method achieved a Hits@50 value of only 0.0188. In the leader-board of results on the different OGB datasets the scores can be seen to vary significantly between different graphs, as such indicating that the difficulty of the link prediction task varies significantly depending on the specific graph and context. Thus, the poor result by the AA method on this graph could be an indication that the patterns of links in the ELN molecule template graph are difficult to learn. On another note, the baselines cannot make use of the vertex attributes, here fingerprints, which could be a potential explanation for the pronounced disparity in performance. However, looking back on the hyperparameter search, the removal of attributes did not harm the performance of the SEAL models to a large extent.

The SEAL model trained on negatives sampled according to the distribution of positives can be seen to have achieved significantly lower results on all metrics in Table 7.6. This is not surprising given that this result is on the test set of randomly sampled negatives with previous results indicating the same trend, as seen in Table 7.4. One interesting observation is however the difference between the link-equity and the vertex-equity metrics. For all other models, both SEAL and baselines, no real difference can be observed between these types of metrics, but for the distribution trained model there is a significant difference. In terms of link-equity AP this model performs even worse than the baselines, but in terms of vertex-equity MAP it is instead much better. Since MAP looks past the degree of vertices, it is reasonable to assume that this deviation could imply that this model has learned something additional of importance regarding vertex's degree. We explore this further in the coming sections.

7.2.2 Finding Optimal Threshold

For some evaluations metrics, a threshold was needed so that a prediction above that value could be considered predictions of a link and a values below could be considered prediction of a non-existing link. The threshold was chosen such that TPR was as high as possible while the FPR was as low as possible, as illustrated for the Voting x5 model in Figure 7.7. Using this method, the threshold was set to 0.85.

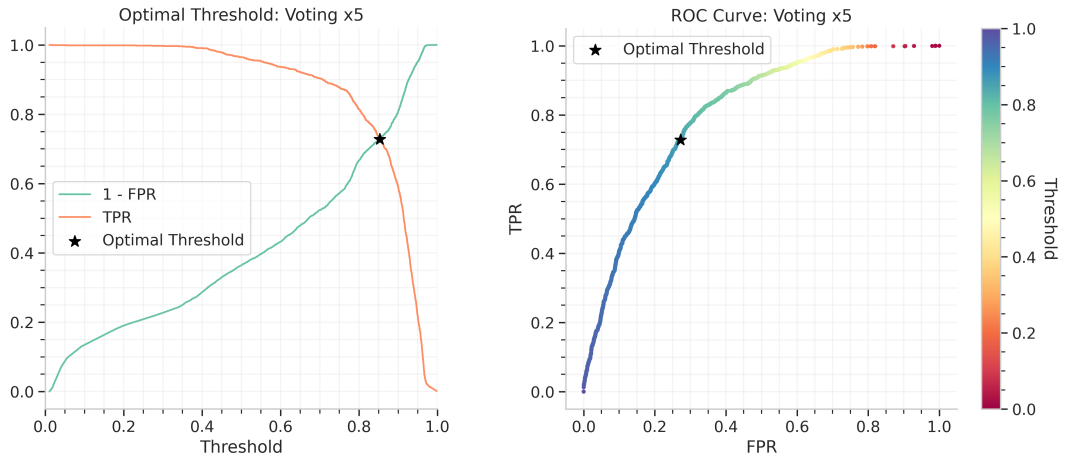


Figure 7.7: Selecting optimal threshold for the Voting x5 model, such that TPR = 1 − FPR.

7.2.3 Analysis of Predictions

Recall that there is no guarantee that a missing link in the graph means that a reaction is not possible. As described in Section 5.3.4, there are however a couple of ways to find more information about the sampled negatives. Most importantly, a subset of the negatives that correspond to reaction templates that describe feasible reactions can be identified according to their presence in the USPTO graph. To see whether the model can predict these links as positives or not, is an indication of the model's ability to generalize and predict missing reactions. On the other hand a link can be verified in terms of whether combining half-templates associated with the source and target vertices are compatible and can generate a syntactically correct reaction template. A rejection of such a match is instead an indication that the reaction is unlikely. Having matching templates is however not a guarantee that a reaction is feasible.

Two of the SEAL models has been chosen as the final models going forward, Random x5 and Voting x5. Voting x5 was included even though Random x5 was slightly better in terms of maximum AUC achieved, since it increased the performance on distributed negatives and structured negatives compared to the model trained only on random negatives. Figure 7.8 shows the cumulative percentage of predicted

instances below a given prediction score, for positive and random negative links in the test set. Also shown are the results on sampled negative links from ELN that are positive in USPTO (226) and links for which no reaction template can be generated (1390). In the figure, the links in USPTO has high FPR, this is good because the ground truth of these links are negative but the model recognizes them as links. The optimal thresholds, θ, as found by the method described above, can be seen in the respective panels. Some additional metrics on the thresholded predictions are presented in Table 7.7.

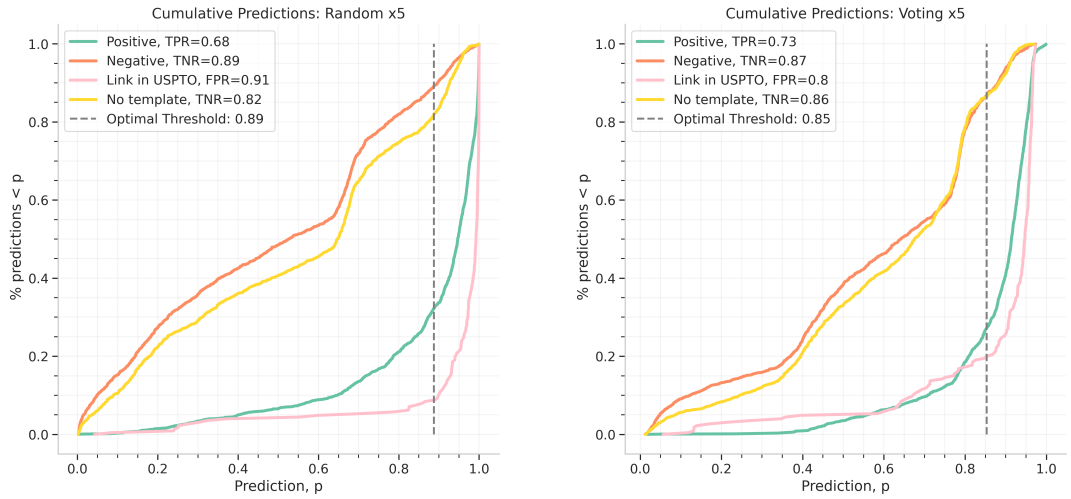


Figure 7.8: Test predictions by the Random x5 model (left) and the Voting x5 model (right). Showing the cumulative number of predictions of each score, for positive and random negative links as well as negative links found in USPTO and negative links for which valid reaction templates cannot be generated.

	Threshold, θ	Accuracy (%)	TPR	FPR	TNR	FNR	% In USPTO $> \theta$	% No Template $< \theta$
Random x5	**0.89**	78.5	0.678	**0.108**	**0.892**	0.322	**91.2**	81.8
Voting x5	0.85	**79.8**	**0.723**	0.132	0.868	**0.272**	79.6	**86.4**

Table 7.7: Evaluation metrics on thresholded predictions from the two best SEAL models, based on their respective optimal thresholds.

Looking at the plots in Figure 7.8 and Table 7.7, one can see that Random x5 model is better at classifying the USPTO links as positive links but slightly worse at recognizing the actual positive links as such. Further, Voting x5 model is better at identifying links which have no reaction template as negative links than Random x5 model. Overall, the two models can separate the classes reasonably well and are especially good at recognizing positive links based on the plots. The thresholds are high, as a result of the models yielding rather high predictions for many of the

negative links. How many of the negative links, for which valid reaction template can be generated, that actually correspond to feasible chemical reactions is unknown at this stage, as the negative links would need to be evaluated one-by-one in an experimental setting. It is therefore not yet possible to make the conclusion that the high FPR corresponds to chemically feasible reaction, even though it could be the case as this is the what we observed for the negative links that were present in USPTO.

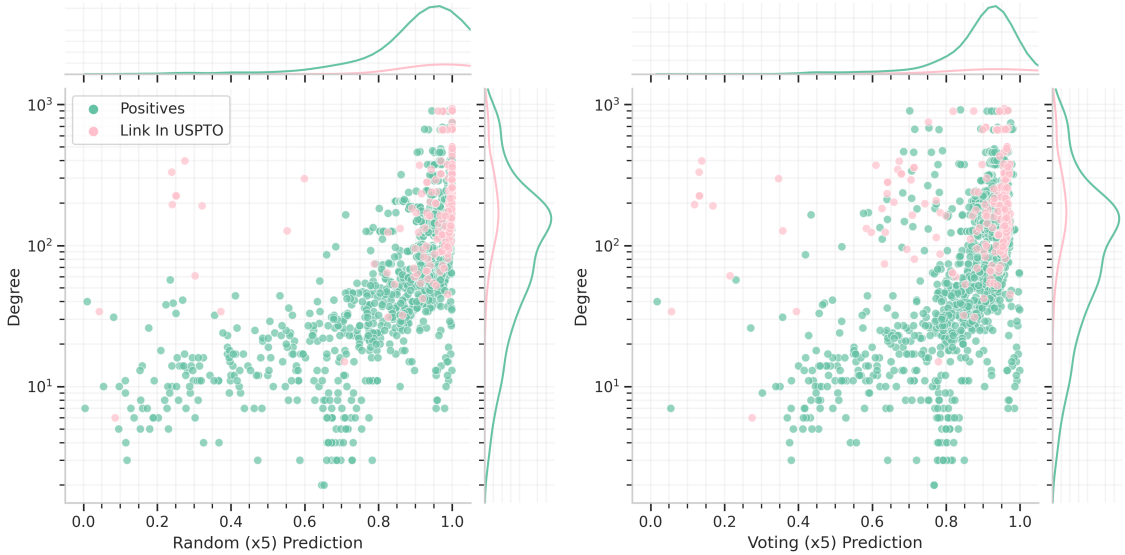


Figure 7.9: Prediction scores versus the total degree of source and target vertices for a given link. Showing positive links together with negative links found in USPTO.

Lastly, the predictions were compared to the total degree of the source and target vertices combined. The motive behind such an analysis was to see if a positive trend could be found, which would indicate that the model had picked up on the fact that more connected vertices are more likely to have other connections. This analysis was done only on the positive class including the negative links present in USPTO, since these are the only certainties in the test data. As can be seen in Figure 7.9, higher probabilities are, in general, more often assigned to links of higher degree vertices by both models. This is however slightly more pronounced by the Random x5 model. A tighter cluster of links with high probability from the model trained on randomly sampled negatives can be seen. The voting model instead seems to have learnt to give positive links higher predictions regardless of the degree. Since this model includes the model trained on distribution negatives, we believe that this further suggests that some valuable information was learnt by training on negative links more similar to the positive links than what is obtained by random sampling.

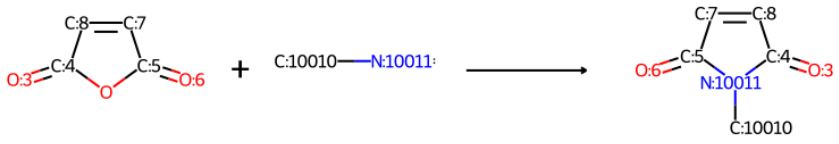
7.2.4 Chemical Feasibility of Top Predictions

For predicted new links in the molecule template graph, multiple reaction templates can be created by combining half-templates of reactions that the molecule templates

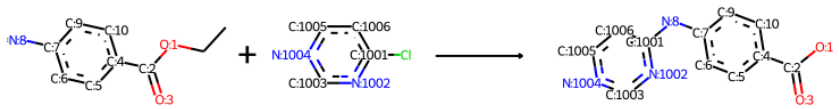
are otherwise involved in, as described in Section 5.3.4. Figures 7.10 and 7.11 show examples of such reaction templates found for the top 10 highest predicted negative links in the test set, by the Random x5 model and the Voting x5 model respectively. For Random x5, only the randomly sampled negatives in the test set were included since the overall performance of the model on the other types of negatives were poor. For Voting x5 however, negatives from all sampling strategies were included. In both cases, links that had already been determined to be real reactions, due to being positive links in USPTO, were exclude since the main interest here was to find novel reactions predicted by the models.

A visual inspection of the predicted reactions seen in Figure 7.11 and 7.10 was done by an experienced chemist within the AstraZeneca team, who concluded that the predicted reactions, with some exceptions, represents reasonable chemistry. The exceptions are reaction number 5 from Figure 7.10 and reaction 6 in Figure 7.11 which are considered infeasible. The reason is an "un-balanced electronic scheme" and the presence of a "penta-valent carbon" in the product molecule, respectively. An important observation is that most of these reactions might have been considered novel by the current graph structures but follow some well established chemistry mechanism.

It is important to stress that the current ELN data used in this study covers chemistry added to the graph between 2003 and 2019. Thus, it is reasonable to assume that some well-known reactions might have evaded the current state of the graph, which greatly influences the exploratory link prediction efforts made within this project. Moreover, a thorough analysis of the reactions and their conditions has to be performed in order to correctly determine the potential novelty of the currently predicted reactions. Nevertheless, our study highlights the need to implement post-processing checks in order to avoid, for example, bad combination of templates leading to infeasible chemistry (e.g. penta-valent carbon).

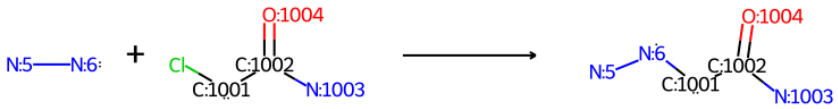


1

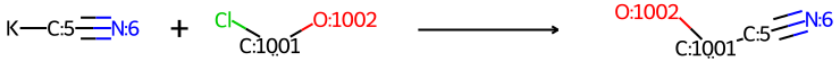


2

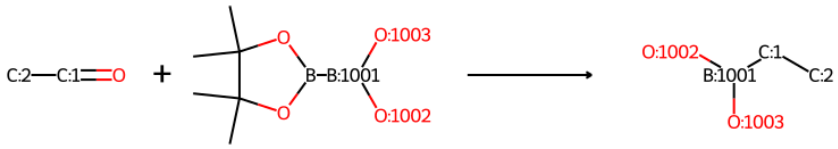
Figure 7.10: Example reaction templates corresponding to the top 10 predictions by the Random x5 model on all randomly sampled negatives in the test set, excluding links already found in USPTO. Showing the first 2.



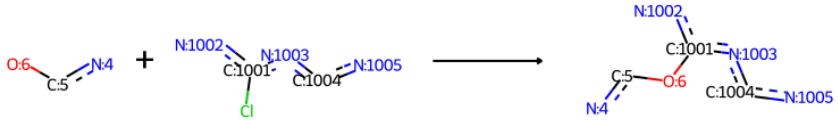
3



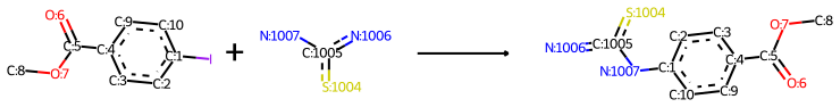
4



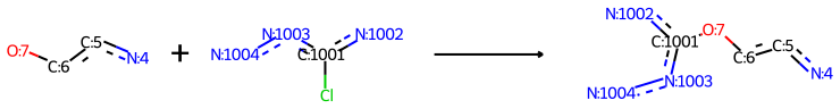
5



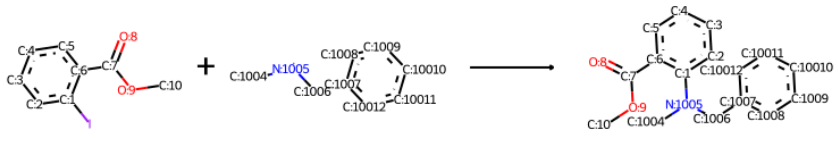
6



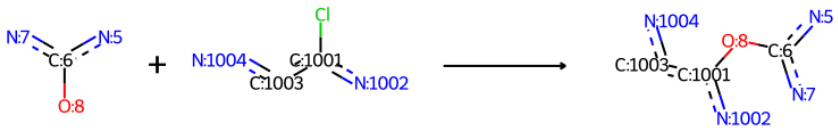
7



8

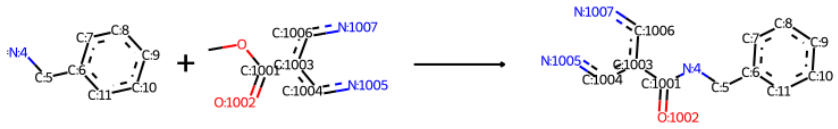


9

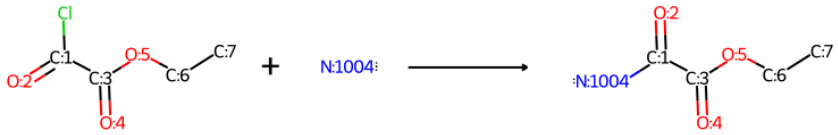


10

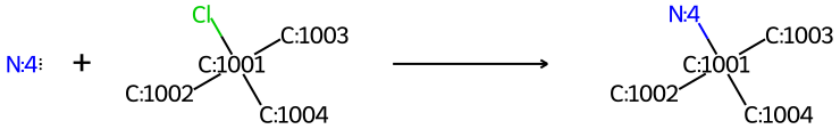
Figure 7.10: Continuation of Figure 7.10, showing the 3-10th highest predictions.



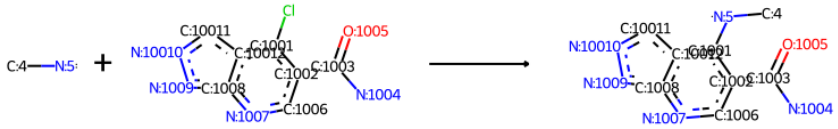
1



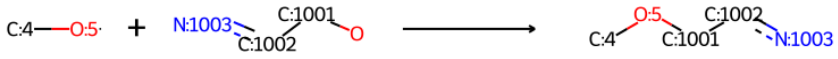
2



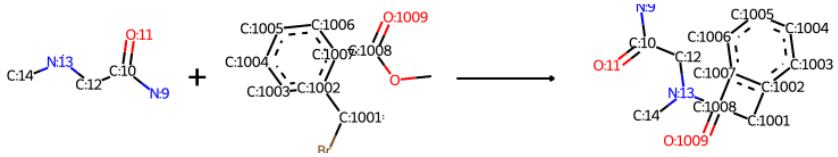
3



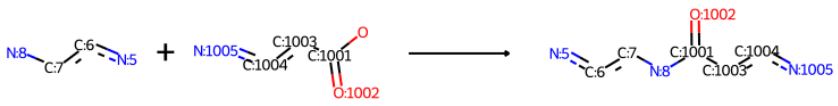
4



5

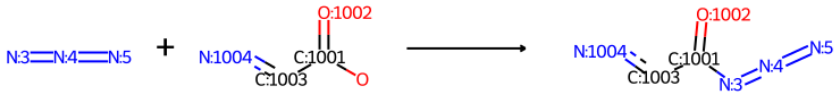


6

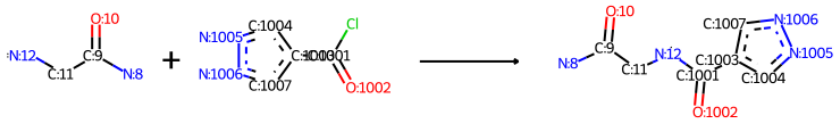


7

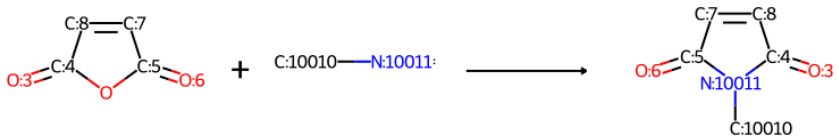
Figure 7.11: Example reaction templates corresponding to the top 10 predictions by the Voting x5 model on all sampled negatives, excluding links found in USPTO. Showing the 7 highest predictions.



8



9



10

Figure 7.11: Continuation of Figure 7.11, showing the 8-10th highest predictions.

8

Conclusions

This thesis has explored the reaction data included in a subset of AstraZeneca's internal knowledge graph of chemical reactions. This was done in two parts. Firstly, by analysing aspects of the graph structure in ELN and comparing them to the findings on other reaction graphs. Secondly, by training the SEAL algorithm to predict new reactions as links in the molecule template space of the knowledge graph.

Important insights about graph structure specifics were initially discovered through the analysis of, and comparison between, the ELN and USPTO graphs. Unlike previously reported results for other reaction graphs, there were no distinct statistical indicators that either of the graphs are truly scale-free. The reason for this was the inconclusive likelihood-ratio tests showing that other heavy-tailed distribution were equally plausible fits. However, power law was consistently one of the best fits, especially compared to the fit to the exponential distribution. A hierarchical nature could be confirmed in both graphs, as also found in previous research [6]. This was concluded from the trends in average degree dependent betweenness and clustering combined with the presence of hubs. Thus, the graphs were still considered to exhibit scale-free behaviors.

Small-world behavior varied between the two graphs. ELN was found to have a rather short average path length as expected both from literature and due to the relatively small size of the network. On the contrary, USPTO had a surprisingly long average shortest path length that could be explained by the presence of some unusually long and isolated synthesis pathways.

In terms of network components, such as core, periphery, and islands, the graphs were overall comparable to previously reported results based on the Beilstein database [5]. In the USPTO case, a core could be identified and confirmed to be made up of building-block molecules. ELN on the other hand, did not exhibit such a core component. The proportions of periphery and island compounds were somewhat consistent for both graphs and as expected the molecular complexity increases by moving from the periphery out to islands. Especially low complexity was found in the hub molecules.

For the purpose of proposing novel reactions through link prediction, hyperparamters, settings and configurations were first optimized through iteratively training with

different settings and finally selecting the optimal case. Increasing the number of hidden channels in the graph convolutions were found to have a positive impact on the model's performance. However, since an increase in this hyperparameters also increased the training time, 32 hidden channels was deemed the most reasonable option for the remaining experiments. Apart from this, 2-hop sub-graphs, dropout between layers and the inclusion of fingerprint attributes to molecule template vertices were found to also be positive for the the model's performance. Increasing the molecule template radius from reaction cites were explored as well, but could not be determined to improve the performance, neither did the addition of normalization or vertex embeddings.

An important aspect of the link prediction in this work was to explore ways to sample the non-existing links, needed for the training and evaluation of the link prediction models. In addition to randomly sampling the negative links, two alternative ways were proposed and evaluated; sampling from the vertex distribution of the positive links or partly so. Models trained on the different sets of links were found to learn differently, but neither of the alternatives outperformed the standard model when evaluated on random links. A voting ensemble model was then constructed, which averaged the results from models trained on the different sampling methods. This model achieved overall higher AUC when evaluated on links sampled according to the different strategies.

Three of the four proposed final models greatly outperformed heuristic baselines, which have previously proven to be competitive with the SEAL model on other graph benchmarks. This spoke both to the superior performance of the suggested models but also to the difficulty of the task in the given graph. Further related results reveled that many of the absent links in this graph could in fact be positive links, according to chemical feasibility, thus underscoring an uncertainty in the negative class. This was deemed to likely affect both the training and the evaluation of the models negatively, and was one of the reasons alternative sampling methods were explored.

Additional analyses to verify the proposed links were performed, such as determining whether negative links were positive links in USPTO and if their molecule templates could be combined into valid reaction templates. Based on these measures the performance of the final models could be further distinguished. For this it was evident that the voting ensemble model had learnt additional information compared to the model trained only on randomly sampled negative links.

In conclusion, the ELN graph was found to have most of the graph theoretical properties seen in other real-world networks, and specifically other chemical reaction networks with some exceptions. These exceptions were likely due to the drug discovery context of the reaction graph. Link prediction using GNNs were shown to achieve satisfactory performance in the graph, yet with further development improved performance should be possible as shall be discussed in the remaining section.

9

Future Works

There are many potential avenues for exploring the existing reaction knowledge graph. For example, additional analyses of graph structure, its reaction classes, temporal evolution as well as relationships of all of these aspects.

Further improvements and evaluation of the link prediction model would likely be beneficial for the sake of predicting novel reactions. Due to the limited optimization that was possible in the scope of this thesis, the authors believe that performance of the link prediction model could be improved with further experimentation. Note that the hyperparameter optimization was done only for molecule templates of radius 1 and for the random negative sampling strategy. A first priory of future research should thus focus on optimizing the other scenarios individually, since it is not implied that these scenarios should have the same settings. Regarding molecule template radius and the alternative negative sampling methods, no hyperparameter optimization was performed specifically for each of these settings. This should be done in the continuation of the research, since these options reasonably could have benefited from a different set of hyperparameters.

Regarding the negative sampling methods, one could imagine additional ways to incorporate the different types of negatives. Presented in this thesis were models trained exclusively on either version. Another option explored was to train first on randomly sampled ones and then on the more difficult option. This did not show a clear improvement and was excluded from the results, but more elaborate such training procedures have the potential to improve the result. One possible such option would have been to successively include more and more of the "harder" negatives in the training or switching back and forth between different strategies.

Lastly, it would also be a good idea to explore the chemical feasibility of novel predictions in greater detail. This could give a better understanding of the actual performance of the model as well as the differences in what the model learns when trained on randomly sampled negatives compared to when trained on other kinds of negatives. As such this would better evaluate the negative sampling methods in relation to each other. Finally, doing link prediction directly on molecules would be a more direct means to propose chemical reactions.

Bibliography

[1] O. Engkvist *et al.*, "Computational prediction of chemical reactions: current status and outlook," *Drug discovery today*, vol. 23, no. 6, pp. 1203–1218, 2018.

[2] S. Johansson *et al.*, "AI-assisted synthesis prediction," *Drug Discovery Today: Technologies*, vol. 32, pp. 65–72, 2019.

[3] J. N. Wei, D. Duvenaud, and A. Aspuru-Guzik, "Neural networks for the prediction of organic chemistry reactions," *ACS central science*, vol. 2, no. 10, pp. 725–732, 2016.

[4] M. H. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic AI," *Nature*, vol. 555, no. 7698, pp. 604–610, 2018.

[5] B. A. Grzybowski, K. J. Bishop, B. Kowalczyk, and C. E. Wilmer, "The 'wired' universe of organic chemistry," *Nature Chemistry*, vol. 1, no. 1, pp. 31–36, 2009.

[6] P.-M. Jacob and A. Lapkin, "Statistics of the network of organic chemistry," *Reaction Chemistry & Engineering*, vol. 3, no. 1, pp. 102–118, 2018.

[7] M. H. Segler and M. P. Waller, "Modelling chemical reasoning to predict and invent reactions," *Chemistry–A European Journal*, vol. 23, no. 25, pp. 6118–6128, 2017.

[8] D. M. Lowe, "Extraction of chemical structures and reactions from the literature," Ph.D. dissertation, University of Cambridge, 2012.

[9] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[10] I. Daylight Chemical Information Systems, "Daylight theory manual," https://www.daylight.com/dayhtml/doc/theory/, 2011 (accessed January 22 2021).

[11] L. David, A. Thakkar, R. Mercado, and O. Engkvist, "Molecular representations in AI-driven drug discovery: a review and practical guide," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–22, 2020.

[12] A. Thakkar, T. Kogej, J.-L. Reymond, O. Engkvist, and E. J. Bjerrum,

"Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain," *Chemical science*, vol. 11, no. 1, pp. 154–168, 2020.

[13] G. Benkö, C. Flamm, and P. F. Stadler, "A graph-based toy model of chemistry," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 4, pp. 1085–1093, 2003.

[14] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[15] F. Lovering, J. Bikker, and C. Humblet, "Escape from flatland: increasing saturation as an approach to improving clinical success," *Journal of medicinal chemistry*, vol. 52, no. 21, pp. 6752–6756, 2009.

[16] J. Clayden, G. Nick, and S. Warren, *Organic Chemistry*, 2nd ed. Oxford University Press, 2012.

[17] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings," *Advanced drug delivery reviews*, vol. 23, no. 1-3, pp. 3–25, 1997.

[18] H. van de Waterbeemd, G. Camenisch, G. Folkers, J. R. Chretien, and O. A. Raevsky, "Estimation of blood-brain barrier crossing of drugs using molecular size and shape, and h-bonding descriptors," *Journal of drug targeting*, vol. 6, no. 2, pp. 151–165, 1998.

[19] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[20] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[21] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[22] K. J. Bishop, R. Klajn, and B. A. Grzybowski, "The core and most useful molecules in organic chemistry," *Angewandte Chemie International Edition*, vol. 45, no. 32, pp. 5348–5354, 2006.

[23] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[24] J. Alstott, E. Bullmore, and D. Plenz, "powerlaw: a Python package for analysis of heavy-tailed distributions," *Public Library of Science*, vol. 9, no. 1, p. e85777, 2014.

[25] A. D. Broido and A. Clauset, "Scale-free networks are rare," *Nature communications*, vol. 10, no. 1, pp. 1–10, 2019.

[26] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.

[27] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi, "The pursuit of hubbiness: analysis of hubs in large multidimensional networks," *Journal of Computational Science*, vol. 2, no. 3, pp. 223–237, 2011.

[28] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[29] M. E. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," *Proceedings of the national academy of sciences*, vol. 99, no. suppl 1, pp. 2566–2572, 2002.

[30] J. M. Granda, L. Donina, V. Dragone, D.-L. Long, and L. Cronin, "Controlling an organic synthesis robot with machine learning to search for new reactivity," *Nature*, vol. 559, no. 7714, pp. 377–381, 2018.

[31] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[32] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[33] S. Li, J. Huang, Z. Zhang, J. Liu, T. Huang, and H. Chen, "Similarity-based future common neighbors model for link prediction in complex networks," *Scientific reports*, vol. 8, no. 1, pp. 1–11, 2018.

[34] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[35] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[38] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[39] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations*

(ICLR), 2017.

[40] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug discovery today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[41] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[42] K. Zhou, Y. Dong, W. S. Lee, B. Hooi, H. Xu, and J. Feng, "Effective training strategies for deep graph neural networks," *arXiv preprint arXiv:2006.07107*, 2020.

[43] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations (ICLR)*, 2018.

[46] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[47] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[48] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Revisiting Graph Neural Networks for Link Prediction," *arXiv preprint arXiv:2010.16103*, 2020.

[49] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[50] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 751–782, 2015.

[51] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[52] G. Crichton, Y. Guo, S. Pyysalo, and A. Korhonen, "Neural networks for link prediction in realistic biomedical graphs: a multi-dimensional evaluation of graph embedding-based approaches," *BMC Bioinformatics*, vol. 19, no. 1, pp.

1–11, 2018.

[53] P. Goyal and E. Ferrara, "Graph Embedding Techniques, Applications, and Performance: A Survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[54] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014. [Online]. Available: http://figshare.com/articles/graph_tool/1164194

[55] "Rdkit: Open-source cheminformatics," http://www.rdkit.org.

[56] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[57] M. Zhang, "SEAL_OGB – An Implementation of SEAL for OGB Link Prediction Tasks," https://github.com/facebookresearch/SEAL_OGB, 2020 (accessed March 15 2021).

[58] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015.

[59] "OGB: Leaderboards for Link Property Prediction," https://ogb.stanford.edu/docs/leader_linkprop/, (accessed May 21 2021).

A

Additional Graph Analysis Results on Molecule Graph

This appendix presents some additional results regarding the graph analysis on, and comparison between, ELN and USPTO.

Table A.1 and A.2 contains the results from fitting the molecule version of the graphs to power law and the likelihood-ratio tests.

	Log-normal		Positive Log-normal		Truncated Power Law		Exponential		Stretched Exponential	
	R	p	R	p	R	p	R	p	R	p
ELN										
Total	-0.12	0.14	105.05	4.78 e-10	-0.05	0.75	2675.66	2,47e-18	28.79	3.78e-4
In	-0.33	0.53	-0.33	0.53	-0.30	0.44	-0.32	0.36	-0.38	0.51
Out	-5.21	0.09	244.99	3.45e-16	-2.62	0.02	6057.21	4.24e-27	31.60	0.013
USPTO										
Total	-0.02	0.93	186.86	7.54e-18	-1.31	0.11	8967.82	4.45e-65	69.52	3.56e-08
In	-612.30	6.19e-310	10386.72	0.00	1.05e-04	0.99	18825.01	1.16e-162	15771.15	6.00e-08
Out	-4.52	0.08	827.53	4.83e-61	-5.83	6.39e-04	30533.91	4.33e-113	148.45	5.73e-12

Table A.1: Result of comparing power law fit to other distributions by log-likelihood test. This result is from using the molecule graph. Positive R indicates that a power law is a better fit to the data and negative R indicates that the distribution in the table describes the data more accurate. The p-value is the significance of the R value. If $p > 0.1$ the direction of R is not reliable.

	ELN		USPTO	
	γ	$k_{\min}$	γ	$k_{\min}$
Total	2.37	6	2.19	14
In	8.90	5	6.09	2
Out	2.23	3	2.11	5

Table A.2: Estimated parameters for fitting the degree distributions to power law, presenting the obtained power law exponents, γ, as well as the optimal minimum degree used, $k_{\min}$. This result is from using the molecule graph.

			Drug-likeliness			
	# Molecules	MW	#HBA	#HBD	cLogP	PSA
ELN						
Full	233,840	367.68±3.2e-01	5.06±5.2e-03	1.27±2.3e-03	3.04±3.6e-03	76.34±7.8e-02
Islands	72,279	401.02±6.2e-01	5.48±9.8e-03	1.4±4.7e-03	3.4±7.0e-03	83.15±1.6e-01
Periphery	161,561	352.77±3.6e-01	4.87±6.0e-03	1.22±2.6e-03	2.88±4.1e-03	73.29±8.7e-02
Hubs	47	107.0±8.3e+00	1.7±1.8e-01	0.7±1.1e-01	0.49±1.6e-01	30.09±2.7e+00
USPTO						
Full	1,372,550	359.82±1.1e-01	4.55±1.9e-03	1.07±8.8e-04	3.5±1.6e-03	68.27±2.9e-02
Islands	173,777	389.89±3.5e-01	4.88±5.9e-03	1.28±3.0e-03	3.77±5.1e-03	75.59±9.5e-02
Periphery	1,197,495	355.61±1.2e-01	4.5±2.0e-03	1.04±9.1e-04	3.46±1.7e-03	67.22±3.0e-02
Core	1,278	216.34±2.2e+00	2.84±4.3e-02	0.97±3.0e-02	1.97±4.2e-02	51.57±7.9e-01
Hubs	523	129.56±2.6e+00	1.74±5.1e-02	0.62±3.3e-02	0.94±5.3e-02	29.71±8.4e-01

Table A.3: Molecular descriptors related to their drug-likeliness, as in Table 6.5. All of these values are also well within the restriction by the Lipinski's Rule-of-Five for drug-likeliness, but note that the values shown are only averages over each network structure.

B

Graph Analysis on ELN Molecule Template Graphs

In this appendix the corresponding graph analysis is presented on the part of the ELN graph containing molecule templates and their reacts relation.

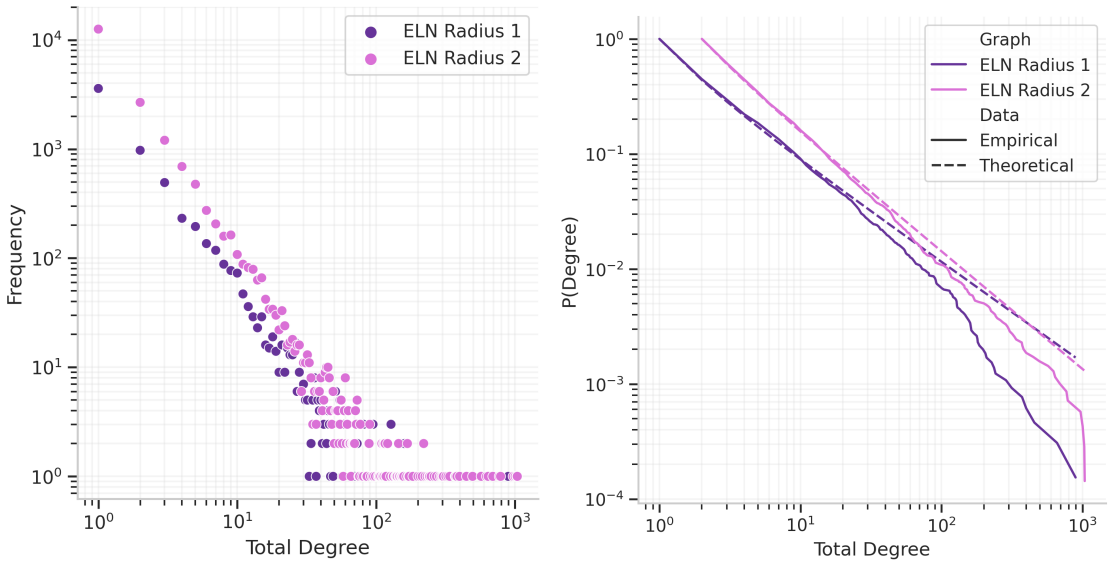


Figure B.1: Degree histogram for total degree, left, and fit to power law, right, for ELN graphs with molecule templates. For ELN Radius 1 $k_{\min} = 1$ was optimal with $\gamma = 1.87$ and for ELN Radius 2 $k_{\min} = 2$ was optimal with $\gamma = 2.02$..

	Log-normal		Positive Log-Normal		Truncated Power Law		Exponential		Stretched Exponential	
Total	R	p	R	p	R	p	R	p	R	p
ELN Radius 1	-20.01	0.00	283.50	0.00	-20.96	0.00	4431.74	0.00	-21.15	0.00
ELN Radius 2	-9.24	0.01	179.98	0.00	-8.45	0.00	4436.26	0.00	-2.83	0.70

Table B.1: Result of comparing power law fit to other distributions by likelihood-ratio tests for the ELN graphs of molecule templates.

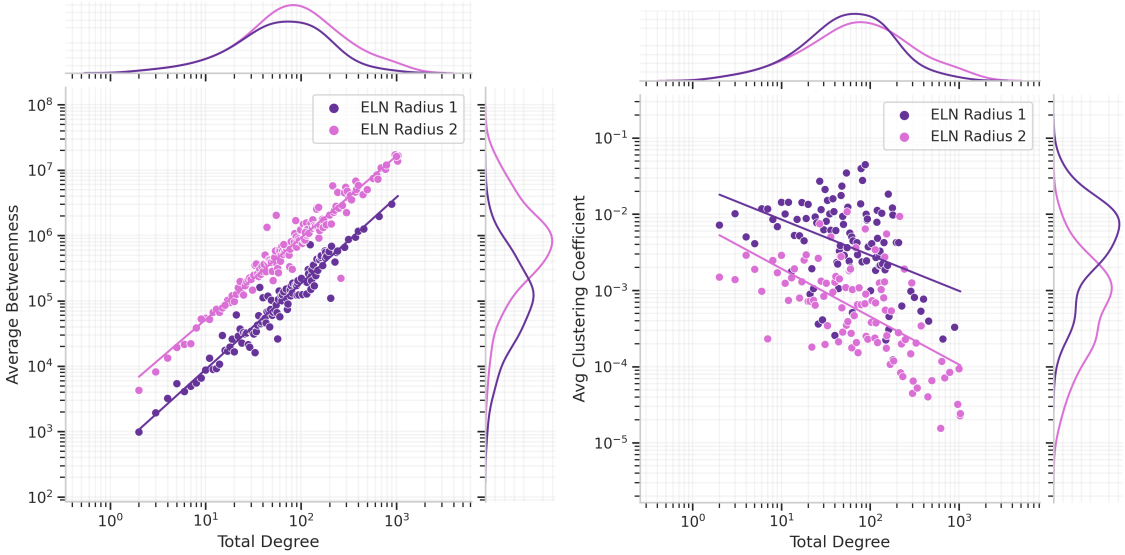


Figure B.2: Average degree dependent betweenness, left, and local clustering, right, for the ELN graphs of molecule templates. The regression lines in the clustering plot has slopes -0.63 for radius 1 and -0.47 for radius 2.

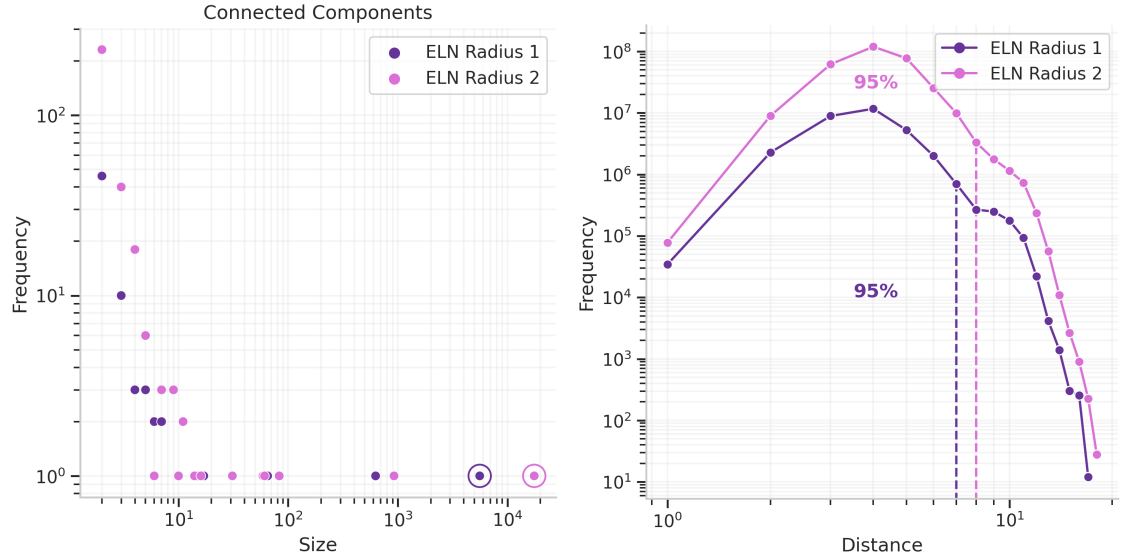


Figure B.3: Histograms over CC sizes in the ELN molecule template graphs in the left panel and distance histograms in the right panel. The average shortest path length in ELN Radius 1 was 4.07 and in ELN Radius 2 it was 4.37.

C

Full Set of Learning Curves

The following appendix shows additional learning curves of all runs from different hyperparameter settings, radius tests and negative sampling strategies.

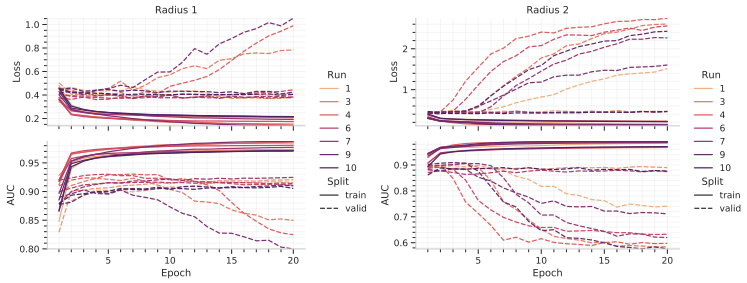


Figure C.1: Extension to the comparison between molecule template radius 1 and 2. Showing each run on the ten random folds used in the cross-validation.

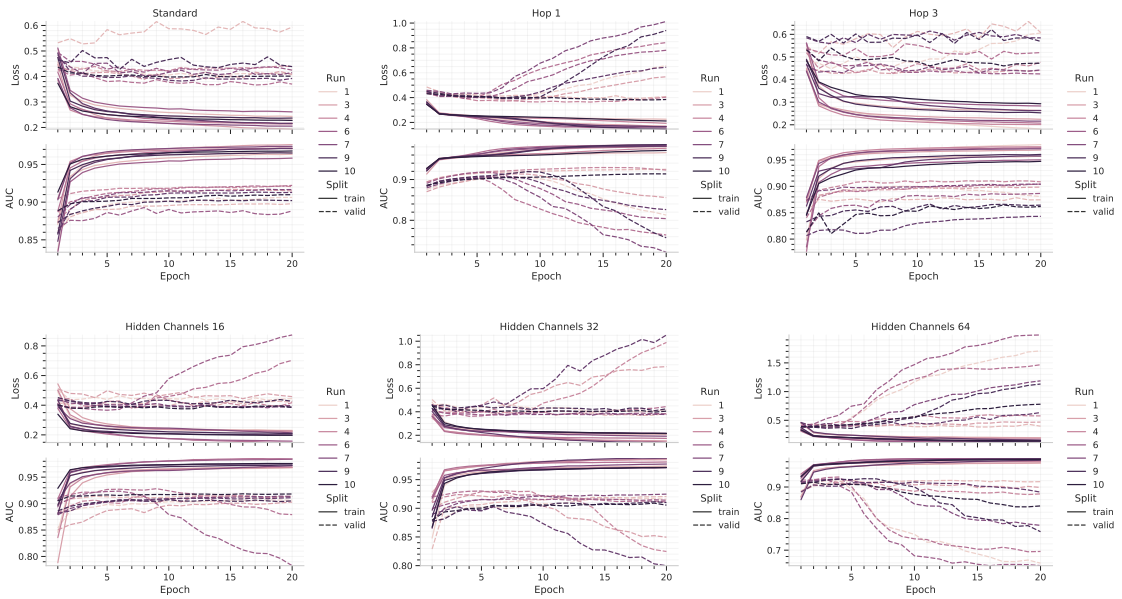


Figure C.2: Extension to the hyperparameter search for cases of hidden channels and number of hops. Showing each run on the ten random folds used in the cross-validation.

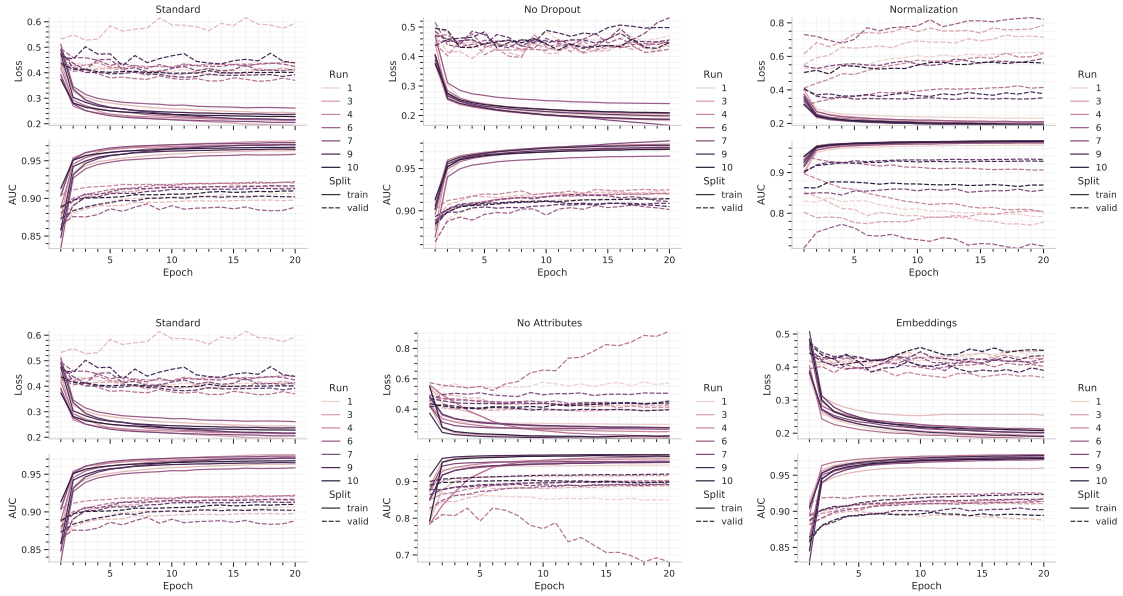


Figure C.3: Extension to the hyperparameter search for cases of GNN modifications and vertex features. Showing each run on the ten random folds used in the cross-validation. The standard case has been duplicated from Figure C.2 for easier comparison.

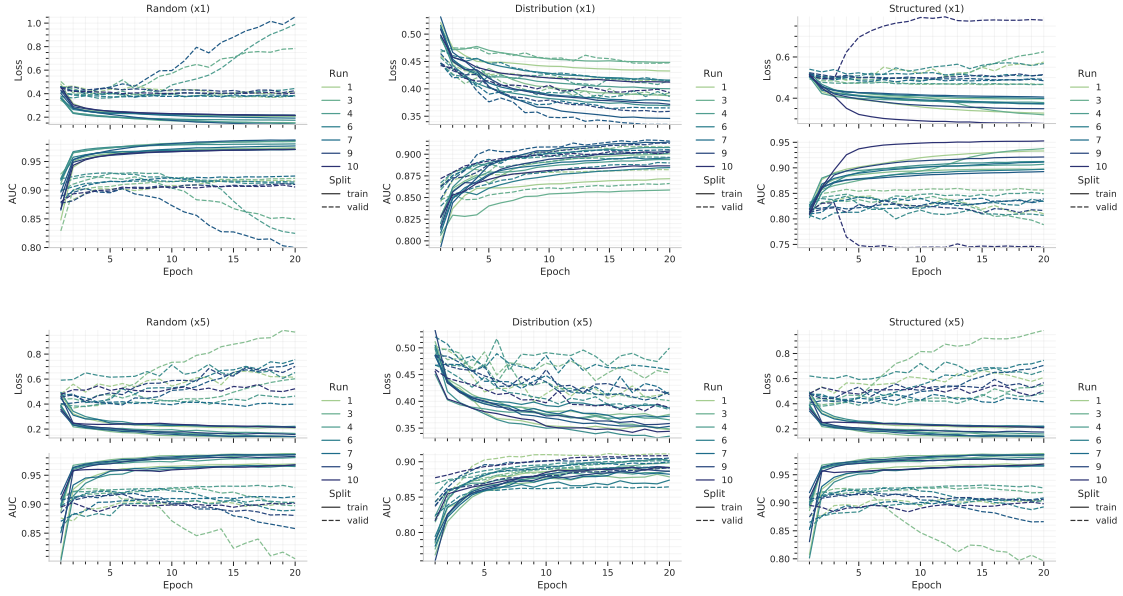


Figure C.4: Extension to the comparison between negative sampling strategies. Showing each run on the ten random folds used in the cross-validation.