Understanding the Role of Background Knowledge in Predictions

Ecotoxicological Effect Prediction

Nils Petter Opsahl Skrindebakke

Thesis submitted for the degree of
Master in Informatics: Programming and System Architecture (Software)
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2020

Understanding the Role of Background Knowledge in Predictions

Ecotoxicological Effect Prediction

Nils Petter Opsahl Skrindebakke

Understanding the Role of Background Knowledge in Predictions

http://www.duo.uio.no/

Printed: X-press printing house

Abstract

Machine Learning (ML) models have proven to perform well in a broad range of prediction challenges. However, ML models require discriminating data to learn patterns and be capable of performing predictions. In this master thesis, we implement a ML model for ecotoxicology effect prediction. To connect the various chemicals and provide the model with a data landscape it can discover patterns from, we utilize a Knowledge Graph (KG). However, KGs are extensive and inevitably contain undiscriminating and noisy data. Thus, we aim to better the performance by exploring methods of filtering and prioritizing the triples in the KG.

We have created several algorithms for filtering and prioritizing the KG. Based on the assumption that discriminating triples connects to either toxic or non-toxic chemicals, the algorithms crawls the graph form each chemical, and scores the triples visited based on the toxicity of the chemical. Because the graph is connected (*i.e.*, there is a path between every pair of vertices), we demonstrate various ways to crawl and score the triples in the graph.

The results show that our approaches outperform the baseline method of using the entire KG equally. Further, we discuss the possibility of these approaches to provide explainability for the predictions.

Contents

List of Figures

Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

AUC Area Under Curv.

BLM Baseline Method.

ChEBI Chemical Entities of Biological Interest.

CID Compound Identifier.

CNN Convolutional Neural Networks.

CORL Combining Ontology and Reinforcement Learning.

DIC Descending Influence Crawl.

DL Deep Learning.

DRC Directed Crawl.

ECOTOX Ecotoxicology knowledgebase.

EPN O-Ethyl O-(4-nitrophenyl) phenylphosphonothioate.

GLUE General Language Understanding Evaluation.

HolE holographic embeddings.

IBM International Business Machines.

KENN Knowledge Enhanced Neural Networks.

KG Knowledge Graph.

KR Knowledge Representation.

LC Lethal Concentration.

LD Lethal Dose.

LSC Limited Step Crawl.

LTN Logic Tensor Networks.

MeSH Medical Subject Headings.

ML Machine Learning.

NIVA Norwegian Institute of Water Research.

OCM Only CID-mapped.

ON One Hot.

OWL Web Ontology Language.

RAdb Risk Assessment database.

RDF Resource Description Framework.

RDFS Resource Description Framework Schema.

ReLU Rectified Linear Unit.

ROC Receiver Operator Characteristic.

SCR Supplementary Concept Records.

SPARQL SPARQL Protocol and RDF Query Language.

SW Semantic Web.

SWT Semantic Web Technologies.

TCE Triple Context Embedding.

TERA Toxicological Effect and Risk Assessment.

TransE Translating Embeddings.

URI Uniform Resource Identifier.

W3C World Wide Web Consortium.

WA Weighted Average.

WWW World Wide Web.

XAI Explainable Artificial Intelligence.

XML Extensible Markup Language.

ZSC Zero-Shot Classification.

Preface

The journey of working on this thesis has been challenging and demanding, but at the same time, truly inspiring. My understanding of Machine Learning and Semantic Web Technologies have profoundly increased, along with new knowledge on Knowledge Graph embeddings, among other topics.

I have explored new approaches to improve ecotoxicology effect prediction and its explainability. However, there are still many improvements to be made, left out of the scope of this project. I hope this contribution can be of inspiration for future work.

I want to thank my supervisors Ernesto Jimenez-Ruiz and Erik Bryhn Myklebust, for excellent advice and guidance. This work would not have been fulfilled if it were not for your help. I would also like to thank Thea Hvalen Thodesen, my fellow student, for motivating me through this project.

Part I
Preliminaries

Chapter 1
Introduction

Machine Learning (ML) have the past years received much attention. The technology is not new, but with today's computing power, and the access to big data, the industrial applications have become much more relevant. One of the domains, in which ML has shown to be beneficial, is ecotoxicology effect prediction.

Ecotoxicology is the study of toxic chemicals effect on living organisms [80]. Commonly, only a few combinations of chemical-species pairs is examined. The laboratory experiments are laborious and can raise ethical concerns if performed unnecessarily. Hence, it is of great value to predict the consequences of unknown chemical-species combinations from known combinations [55, p.1].

In this exploratory thesis, we implement a ML model for the ecotoxicology domain. The objective is to predict the effect compounds have on particular species. To accomplish this, we need to provide the model with data from which it can learn patterns. To connect the various chemicals with discriminating data, we introduce Knowledge Graphs (KGs).

A significant weak spot in some ML models is the lack of explanations for its prediction. The process is black-boxed, and the user usually gets no indication of how the algorithm decided the output. It can, therefore, be a problem for users to trust the system and make decisions based on something unexplained. Providing explanations could also help the users obtain more knowledge of the data to examine further. As a second challenge, we will explore if explanations could be provided with the help of KGs. However, that it is not in the main focus of this project.

The thesis is divided into ten chapters. In this chapter, we have given a brief introduction and presented the master's thesis project. The following

Chapter 2 will explain the relevant research fields to give the reader the necessary background knowledge, while Chapter 3 will presents related work. In Chapter 4, we will justify the choice of programming language, ML framework, and the models we use. Chapter 5 explains the data used in this project, and the transformation performed on them. Chapter 6 describes the various methods we propose with the concepts behind them. The details on the execution and the results form the experiments are found in Chapter 7 and 8, respectively. Chapter 9 contains discussions on our findings, and lastly, Chapter 10 concludes our work before discussing future directions.

Chapter 2
Background

This thesis incorporates a broad section of technologies in the Artificial Intelligence (AI) research domain. The reader is in this chapter introduced to the essential topics we encounter during this research. The first sections describe the various areas of study. Section 2.1 introduces ecotoxicology, which is the setting of the data sources. Section 2.2 is a more comprehensive part that presents the Semantic Web Technologies (SWT), including RDF and Ontologies. Machine Learning is then represented in Section 2.3, while Section 2.4 and 3 are about combining ML with Knowledge Graphs and relevant projects.

2.1 Ecotoxicology

René Truhaut introduced the term ecotoxicology in the late '60s. The name is derived from two research areas: ecology and toxicology. Ecology is the study of relations between organisms and their soundings, while toxicology is the study of negative impacts chemicals have on living organisms. Ecotoxicology analyses the harmful effects on ecosystems and individuals when exposed to different chemicals [80].

2.1.1 Ecotoxicology and Risk Assessment

The inspiration of this work is the preliminary study in [55] which investigates the use of KG embeddings in ecotoxicological effect prediction. This study is based on the use case at the Norwegian Institute of Water Research (NIVA) where predicted effects are integrated into the Risk Assessment database (RAdb)[1]. RAdb is a system for performing ecological risk assessment case

[1]NIVA RAdb: https://www.niva.no/en/projectweb/radb

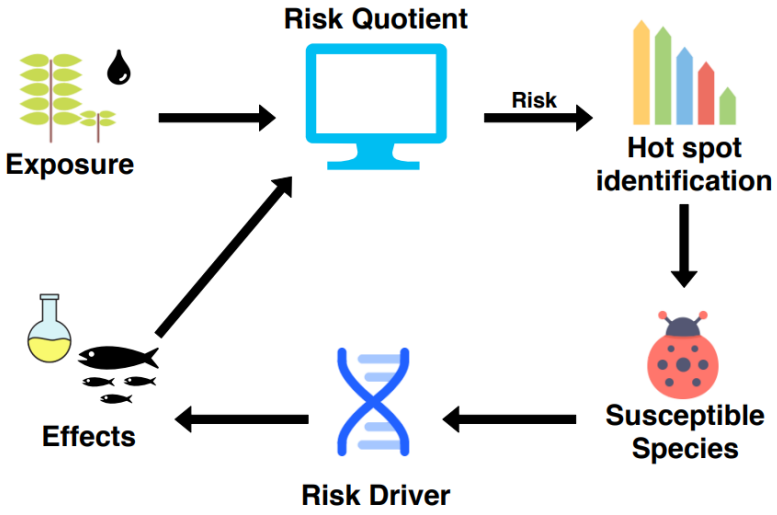


Figure 2.1: An ecological risk assessment pipeline, simplified. Reprinted with permission [54].

studies concerning agricultural/industrial runoff into lakes and fjords.

The ecological risk for an ecosystem can be seen as

$$\text{Fate} + \text{Effects} = \text{Hazard assessment}$$
$$\text{Hazard} + \text{Exposure} = \text{Risk}$$

The ecological risk assessment is a combination of the hazard of a substance and an estimate of the environmental exposure. The hazard is evaluated from the effect of the chemical and fate (*i.e.*, transport, transformation and breakdown) [37].

> "Prediction models for combined toxicity and cumulative risk are used to assess the effect of chemical mixtures and multiple stressors, whereas population and community modelling will be implemented for impact assessment where traditional laboratory data cannot be easily acquired."
> - NIVA's Computational Toxicology Program team [36].

This citation encompasses the goal of risk assessment as a field. In other words, prediction models are used to assess chemical effects where it can be hard to collect laboratory data.

As we can see in Figure 2.1, the risk assessment pipeline is a circular process. It has two primary sources: effect and exposure. The exposure is

data collected from the environment, and the effects are the hypothesis tested in the laboratory. The risk is used to to create new hypothesises for further testing, completing the circle [55].

2.1.2 Ecotoxicological Endpoints

To compare the toxicity between different compound, it is necessary with standard units of measure. The different units are referred to as endpoints. The mortality rate is most commonly used and measured at time intervals until it becomes constant. Two frequently used endpoints are LD50 and LC50. LD50 expresses the lethal dose (ingested) for 50% of the test population, while LC50 indicates the lethal concentration (in the environment) for 50% of the test population [55]. LD50 is often measured by g/KG, and LC50 by g/L or mol/L.

In this project, we will look at how chemicals affect the fish called fathead minnow. LC50 was used to measure the data as this is more suitable for testing on fish. LC50 consists of mixing the chemicals into the water, in contrast to LD50, where the organism is feed directly.

2.2 Semantic Web Technologies

A concept particular central to this paper is the Semantic Web (SW). It was introduced in the late '90s by Berners-Lee, the man behind the World Wide Web (WWW) [23]. The idea was to make the data on the web readable, not only for humans but also for computers. It could have given significant benefits by enabling reasoning on the combination of data from sources all over the world. It did not catch on in the beginning, but the appliance of this technology in closed systems have become more and more relevant to the industry in the last years, especially in the petroleum and healthcare industries.

Tim Berners-Lee is also inventor and director of a global community that sets web standards, called The World Wide Web Consortium (W3C). W3C's Semantic Web Technologies (SWT) stack includes RDF, RDFS and OWL. Figure 2.2 shows how the different elements build on top of each other.

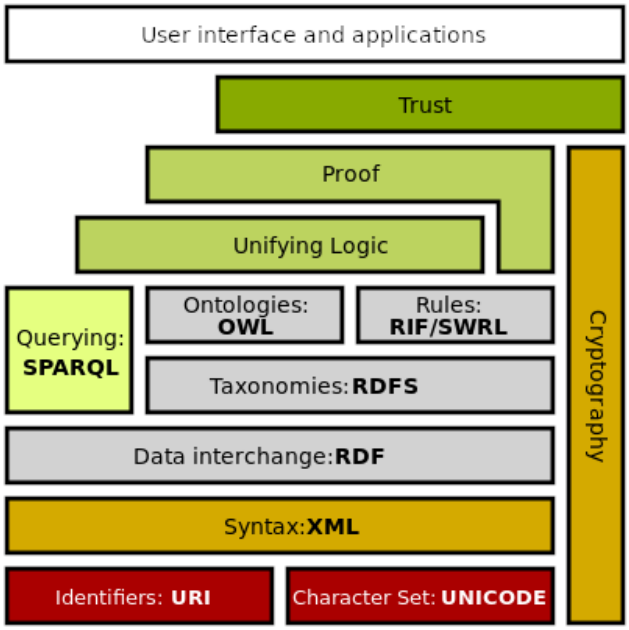


Figure 2.2: W3C's SWT stack [77].

2.2.1 RDF

It was a challenge to structure the heterogeneous data from the numerous sources as machine-readable. The solution was to store all the information in triples. The W3C has a standardized framework for this, called Resource Description Framework (RDF) [41]. The first instance is the subject that is described; the second is the predicate describing what relation it has to the last instance, the object. Using this technique, we can describe most things. An example of this is the triples below.

:Alice foaf:knows :Bob

The subjects and objects are perceived as nodes in a graph, where the predicates are directed edges between them. Customarily, the nodes and the predicates are in the form of Uniform Resource Identifiers (URIs) that represents the unique class.

However, the nodes can also be blank nodes. Blank nodes are unknown classes that are used between known classes. This enables, for instance, to describe the grandparent relation, without knowing the parent of the child. Consider the fact: "Alice has a parent that has a parent named Bob". Bob is in other word, Alice's grandparent. We can demonstrate this by using ◯ to represent a blank node in the graph.

:Alice fam:hasParent ◯ fam:hasParent :Bob

The objects can also be literals (e.g. numbers, strings, boolean). A literal cannot be a subject nor a predicate. If we would want it to express, for instance, that "10 is a number", we would have to use a resource that represents the number ten.

:Alice foaf:age 10

ex:Ten rdf:type ex:Number

The RDF initially builds over Extensible Markup Language (XML), but as this is hard to read for humans, there exist several syntaxes that compile down to XML. The examples in this chapter has an `N-Triples` syntax.

2.2.2 Ontologies

As one can see in the previous examples, the elements either begins with a colon or a prefix followed by one. The prefix is pointing to a URI of a vocabulary. If there is no prefix before the colon, it refers to the default one. This way, RDF can combine elements from different vocabularies. `FOAF`[2], which contains the predicate `Knows` in the example above, is a vocabulary describing people and their relations.

A synonym for vocabularies is ontologies. Ontologies or vocabularies define, as W3C states: "concepts and relationships used to describe and represent an area of concern" [40]. In other words, it expresses the facts of a specific domain. W3C later specifies that:

> "There is no clear division between what is referred to as `vocabularies` and `ontologies`. The trend is to use the word `ontology` for more complex, and possibly quite formal collection of terms, whereas `vocabulary` is used when such strict formalism is not necessarily used or only in a very loose sense."[40]

Resource Description Framework Schema (RDFS) is a vocabulary that supplies essential components to describe other vocabularies. Properties can,

[2]FOAFs URI: `http://xmlns.com/foaf/0.1/`

for instance, be described with domain and range, declaring what type the subject and object must be. At the same time, `subClassOf` and `subPropertyOf` can define the hierarchy of classes and properties. The triples below specify: A Student is a subclass of person. The predicate `studyAt` expect a person as a subject, and School as an object. We have also expressed the triples with description logic above the triples.

$$\text{Student} \sqsubseteq \text{Person}$$
:Student rdfs:subClassOf foaf:Person .

$$\exists \text{StudyAt}.\top \sqsubseteq \text{Person}$$
:studyAt rdfs:domain :Student .

$$\top \sqsubseteq \forall\ \text{StudyAt.School}$$
:studyAt rdfs:range :School .

OWL is another part of the W3C's SWT stack. It is a vocabulary that describes how the classes are related (e.g. disjointness), cardinality, equality, properties characteristics (e.g. symmetry), and much more. One can, for instance, say that an class cannot be both a person and a school:

$$Person \sqcap School = \emptyset$$
:Person owl:disjointWith :School

The triples in a vocabulary are separated into two groups, TBox and ABox. The triples in Tbox describes concepts relevant for the data, while the triples in ABox contains statements on individual classes [31]. The triples from our previous examples that have the predicates subClassOf, domain, range and disjointWith are typical TBox instances. In contrast, the triples with the predicates age, hasParent and knows are a part of the ABox group.

2.2.3 Knowledge Graphs

Google introduced in May 2012 a new way of searching the web. This activity has previously required considerable human effort by matching the right search words. Applying SWT enables the search engine to distinguish between various objects as things and not only strings. This facilitates the search task,

making it effortless and more comprehensive. The searches are now both more precise and contain more relevant information. Google calls this approach for Knowledge Graph (KG) [71].

Figure 2.3: The Google search of Alan Turning collects relevant information like where he was born, his partner and his education. The text highlighted in blue is not only strings but hyperlinks to other pages. These pages are again connected to other objects, allowing the user to crawl the KG.

The use of the term KG can be traced back to the 1970s. However, the current perception of the term was established after googles publication, followed by other big corporations publications (*e.g.*, Amazon, Facebook, Microsoft) [44]. The technology is not entirely new, and RDF-based KGs have been out for a while [2]. However, the term KG has since been used broadly in the literature, both in the industry and academia. An official definition is not agreed upon, and many suggestions have been proposed [35, 44]. An example of an adopted definition is:

> "A graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities."
> - Inclusive KG definition, Aidan Hogan *et al.* [44].

Many of the definitions are similar to RDF, if not recognizing it as a synonym. The term has also been categorized as a buzzword made by Google for knowledge representation applications [35]. However, we will use the term KG in the following chapters. KG is usually applied when referring to both the vocabulary and the data itself [66] (*i.e.*, both TBox and ABox). It is also descriptive to the way we see the data, as a graph of knowledge, and not only as individual triples.

2.3 Machine Learning

The inspiration for Machine Learning (ML) is from the animal kingdom, where the brain is the organ that enables us to learn. By studying how the brain operates, we can create biologically inspired approaches to mimic these mechanisms, with computers.

The brain is a complex and impressive system. It is made up of hundreds of millions of nerve cells, called neurons. These are interconnected with each other and fires electrical signals to their neighbours. When a signal[3] is received, it will forward this signal if it reaches a certain threshold. The thresholds and how powerful the neuron fires are influencing how the output will be. This explanation is a simplification of how the brain works, and the brain is far more complex [51, p. 39].

McCulloch and Pitt introduced in 1943 a mathematical model where they modelled a simplification of the neuron. It consists of a set of weighted inputs, an adder that sums the input signals and activation function[4] [51, p. 41]. This model is the foundation of many ML algorithms. By setting multiple neurons together in a network, similar to the brain, the system can learn from how strong the nodes are interconnected, by updating weights between then [51, p. 43].

If labelled data are available, weights can be adjusted by looking at the produced output compared to the labelled answer. This action will train the network to fit the real world, and a computer can calculate a classification based on an input. This method is called supervised learning and is what this project will be using.

[3] Can be multiple signals from different neurons.

[4] *e.g.*, a threshold that fires if the summed input is great enough.

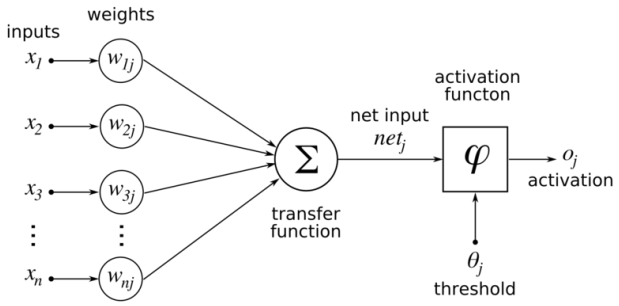


Figure 2.4: McCulloch and Pitt mathematical model of an neuron [28].

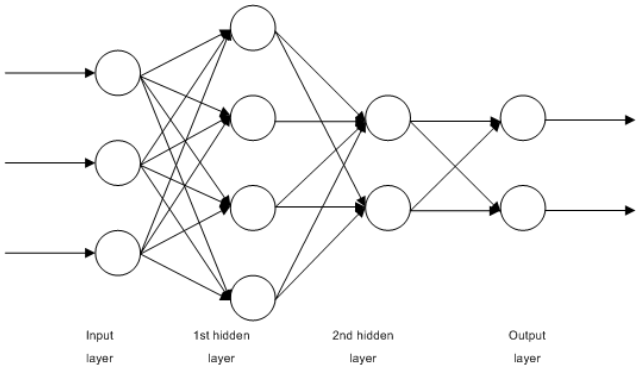


Figure 2.5: Multilayer Neural Network [65].

There are a number of other types of ML, including Unsupervised Learning and Reinforcement Learning. Unsupervised Learning tries to find correlations in the input and categorize them together. Labelled data are not provided in this method, nor is it provided in Reinforcement Learning. Here the algorithm only knows when the output is wrong, and have to explore until it finds the right answer [51, p.6].

2.3.1 Data Preprocessing

"Garbage in, garbage out"
- George Fuechsel

The phrase above is a famous training mantra from the IBM instructor George Fuechsel. He refers to the input data as "garbage in", and the results as "garbage out". The last part of the phrase has further been replaced with "gospel out". The meaning of this is to emphasise the problem of trusting the program excessively. It is essential to provide the models with reliable and beneficial information, for the results to be accurate [47, p. 167]. When

looking at the whole data mining process, the data pre-processing is often the most laborious. This step requires between 60% to 80% of the effort [39]. The step includes, among other things, removal of replicas and missing values, converting to the standard units and mapping the IDs to the same domain when possible.

When the data is of sufficient quality, it must be split into two parts: a training set and a test set. As the names are insinuating, the training set is used to train the model while the test set is used to test the model. It is crucial to never train on the test data. This could give an unrealistic performance, and disguise overfitting. The division is commonly between a 70-30 and 80-20 split but can vary. However, the test set must be large enough to give meaningful results and be representative of the whole set [13].

2.3.2 Overfitting

(a) The real tree [3].

(b) A model of the real tree.

Figure 2.6: Illustration of how a model is a abstraction of the real world.

A model is trained to fit the real world. It will presumably never be an exact representation, but an abstraction with the essential details. The data we train with are never completely representable as they do not hold all the details. Hence the data contains noise: random and insignificant data. The model is only a generalization drawn from this data, like the modelled three (b) is a generalization of the real tree (a) in Figure 2.6.

> "A model overfits the training data when it describes features that arise from noise or variance in the data, rather than the underlying distribution from which the data were drawn. Overfitting usually leads to loss of accuracy on out-of-sample data."
> - Geoffrey I. Webb [84]

Overfitting is, in other words, a term used when the model is trained too well to the training data, and no longer are a general model for the problem. The problem is well-known, and solutions do exist (*e.g.*, regularization).

There is a trade-off for how complex the model should be to provide the best performance. A too simple model will risk not using the data to its full potential. While on the other side, a too complex model will risk having the noise overshadowing the discriminating features.

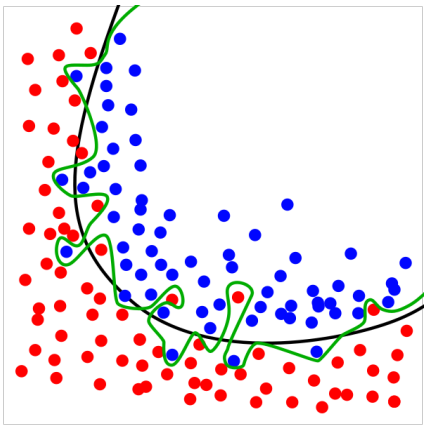


Figure 2.7: The green line represent the classification after overtraining, while the black line represent how the classification is when optimal. The colouring of the dots are representing the real classification of the training data [45].

Regularization L1 and L2

There are two main methods used in regularization, L1 and L2. L1 [1] liquidates the trivial and slightly relevant features setting them to null, by adding a penalty in proportion to the sum of the absolute values to the corresponding weight. In contrast, L2 [1] penalizes with the sum of the squares. This regularization moves the weight in the outline closer to zero.

$$L1 : \lambda \sum_{j=0}^{M} |W_j| \qquad L2 : \lambda \sum_{j=0}^{M} W_j^2$$

The penalties are used with something called regularization parameter. The lambda symbol (λ) represent the parameter in the expressions above. The lower the value is, the less the regularization will influence, and we risk the chance of overfitting. The higher the value, the more influence from the regulation, and we increase the risk of underfitting.

Dropout

Dropout is a method which involves temporarily "dropping" out arbitrary nodes in the network and their outgoing edges, during the training. The nodes will, thus, not evolve too reliant on each other. The creators Hinton *et al.* [72] represents a motivation for this technique from reproduction. One could assume an asexual[5] approach would be superior, as offspring would inherit significant genes directly. These traits could be broken up by sexual reproduction[6], yet advanced organisms have evolved adopting this strategy. A reasonable explanation of this is that the reduction of complex co-adaptations will increase the opportunity of new features to emerge.

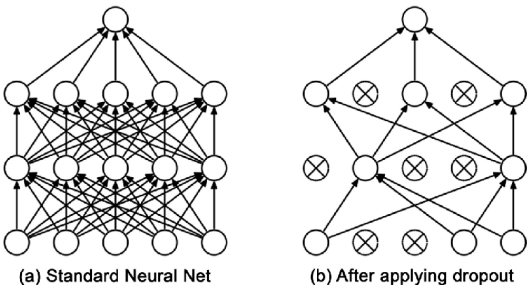


Figure 2.8: Illustration of how a network could look after applying dropout [88].

Early Stopping

The concept of early stopping is to stop the training process when signs to overfitting occur. The approach divides the training data into two divisions: training and validation. After a preferred quantity of training iterations, the algorithm controls the network with the validation data. The initial early stopping technique suggests we stop the training when the error is more significant than the last run. The evolution of the network is, however, not as smooth. Other stopping criteria are, therefore, introduced. One can, among other things, stop when the generalization loss reaches a particular threshold or stop when the error has increased in the last N number of runs [62].

[5]Asexual reproduction: small genetic changes based on only one parent.

[6]Sexual reproduction: combines genetics from two parents with a modest appearance of random mutations.

2.3.3 Activation Functions

The neurons in a network are densely connected. It receives input from multiple sources and shall, based on this, itself send out a signal to the next neuron in the network or the output of the model. In the human brain (the inspiration behind ML), the input to the neurons needs to be of significant strength for it to pass the signal on. The same can be simulated in an artificial neural network. Before the aggregated output is passed on as an input to the next node, it enters an activation function. In this section, we will discuss some of the established activation functions. Plots of them are displayed in Figure 2.9.

The most explicit solution would be to set a threshold, which the sum of the input needs to exceed for the neuron to activate. This Binary function is not smooth and returns either zero or one, if the input is below or above the threshold, respectively.

Another activation function, with a more smooth transition between firing and not, is the Sigmoid function. All values are above zero [59], while still achieving a distinct shift — especially when compared to a linear approach, as we see in Figure 2.9.

However, the activation function Rectified Linear Unit (ReLU) has proven to offer better performance than Sigmoid. On the positive side of the graph, we can see its identical to the linear approach. ReLU eliminates the vanishing gradient problem [59]. The problem occurs when the gradient eliminates the small weights, leaving them stuck at a value close to zero.

A similar function to Sigmoid is the Tanh function. It has a similar distinct shift but returns a result in the range of -1 to 1, in contrast to the Sigmoids range of 0 to 1. Hence, the decision boundary is focused around 0. If we pass in the value 0 to Sigmoid, we would have received the output 0.5, but with Tanh, we will get the value 0.

There are plenty of other activation functions available (e.g. SoftMax, Leaky ReLU, SiLU, ELiSH), but the ones mentioned above are the most relevant for this work.

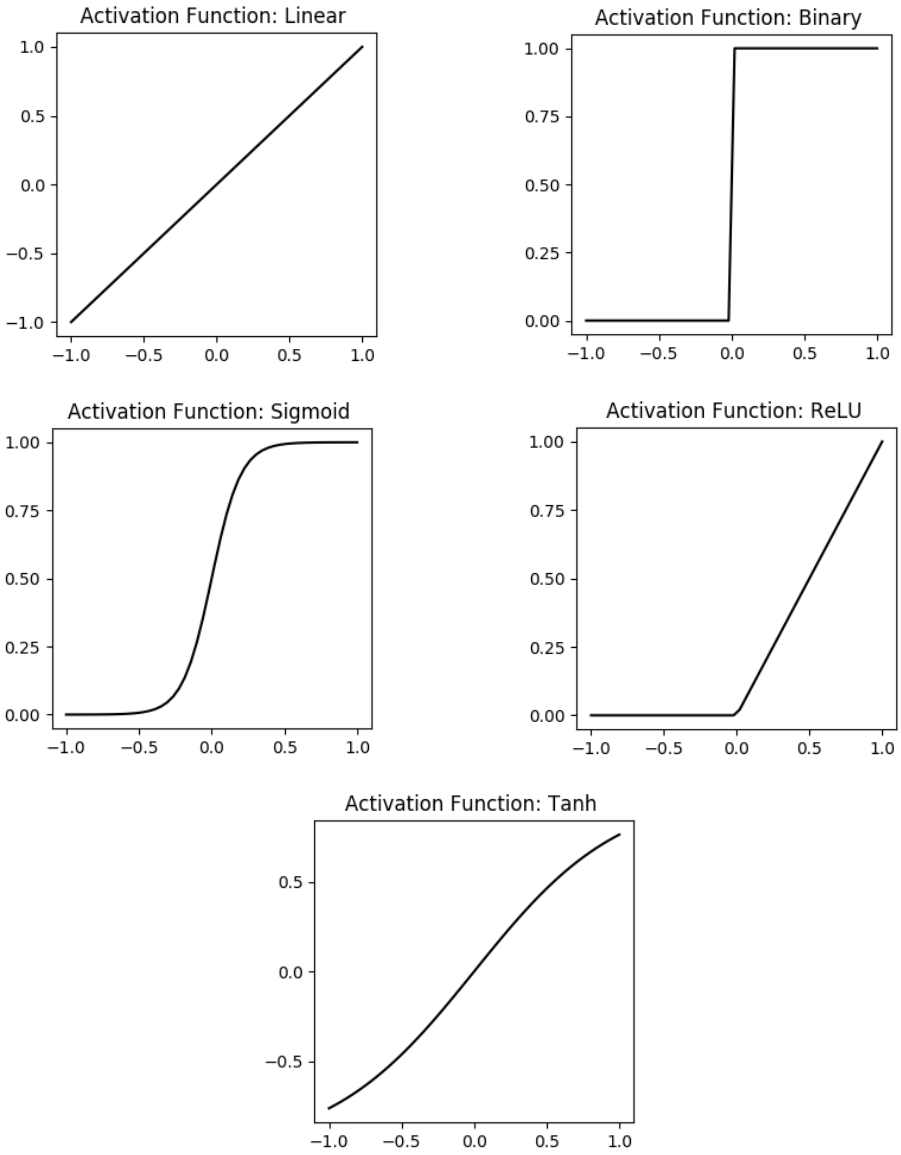


Figure 2.9: Some of the established activation functions.

2.3.4 Deep Learning

Deep Learning (DL) is a type of ML that consists of a neural network with multiple, even up to hundreds, of hidden layers. The division between regular ML and DL are, however, fuzzy and it is no exact size the network needs to exceed for it to be classified as a deep neural network.

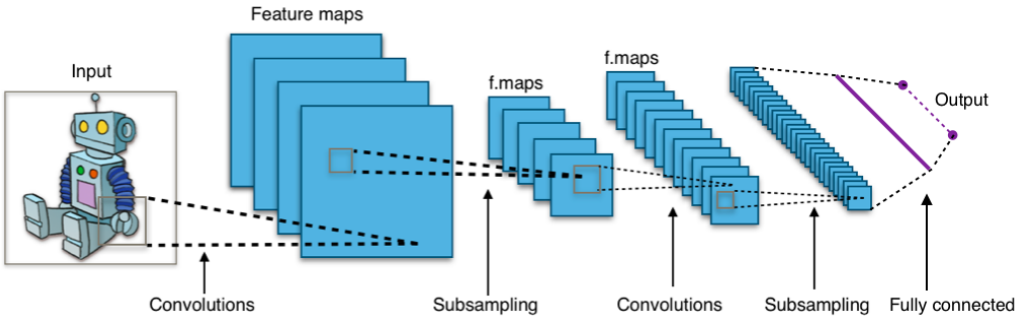


Figure 2.10: A typical Convolutional Neural Networks [76].

DL is not only referring to the size of the network. CNN, short for Convolutional Neural Networks, is again a subcategory of DL. CNN abstract features in separate layers to draw more complicated relations between output and input. The technique applies well to images where the features can be edges, colours or more specific traits. CNNs works best when trained with large data sets [43].

2.3.5 Few/Zero-Shot Learning

Few-Shot Learning is a ML technique that only uses a few labelled samples to train the model, contrary to traditional ML algorithms described in the previous section [85].

Zero-Shot Learning is similar, but here we use no instances of the class we try to predict in the training process. For the class to be completely unseen, and for us to still predict, we need some excess information to connect the classes to other labelled classes.

Figure 2.11, is an example to demonstrate this. We can see that apples share attributes such as consistency and taste with pears and bananas, and colour and shape with grapefruits and the tomatoes. This way, by training the neural network on the seen classes, we can predict, for instance, an image of an unseen apple.

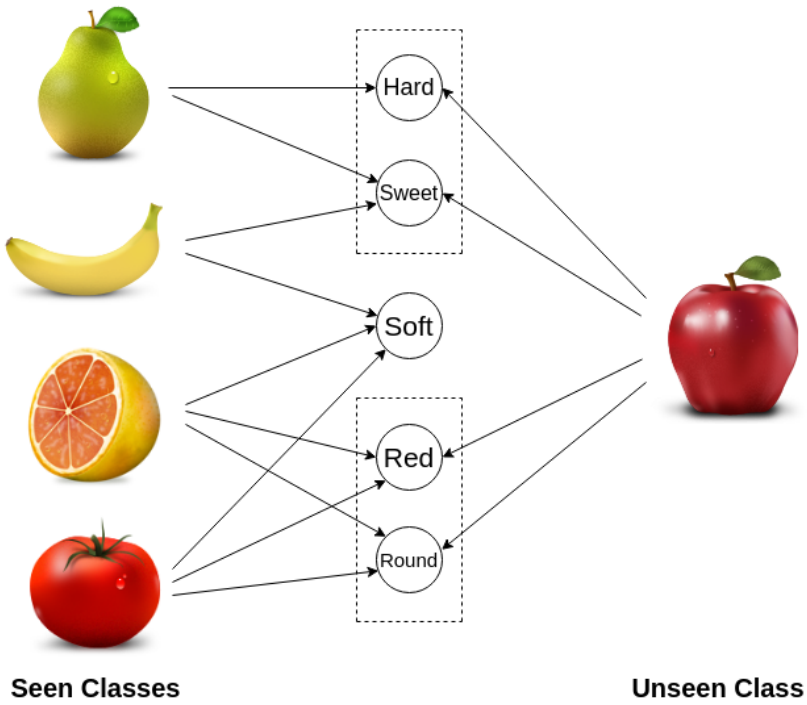


Figure 2.11: The is an illustration of how the unseen class is connected to the seen classes with attributes in common (marked inside a rectangular with dotted lines). Inspiration is drawn from Geng *et al.* [38] example where they recognize a serval for having shard attributes with a cat and a cheetah.

2.4 Combination of Machine Learning and Semantic Web Technologies

We have in the previous sections introduced two distinct types of AI techniques. The first, SWT, involves employing structured data, traditionally modelled by specialists, and reasoning on this data. The second, ML, is concerning learning based on patterns in the data. Combining the two areas can be a powerful tool, and potentially have many applications. Some of which are mentioned in Section 3.

One can unite the techniques in numerous ways. We will classify and introduce some of these various applications in the following paragraphs. We have chosen a task-oriented classification, in contrast to Harmelen and Teijes boxology [78], which focus more on how components are connected. However, inspiration are drawn from their framework.

Knowledge Graph Completion

ML algorithms can accept semantically structured data as input and output, instead of the traditional images and tabular data. This enabling us to learn new triples based on patterns in existing triples, that we might not obtain from reasoning. Inspired by word2vec [53], recent graph completion solutions usually uses graph embeddings to solve this [69, 78]. KG embedding is explained in Chapter 2.4.1.

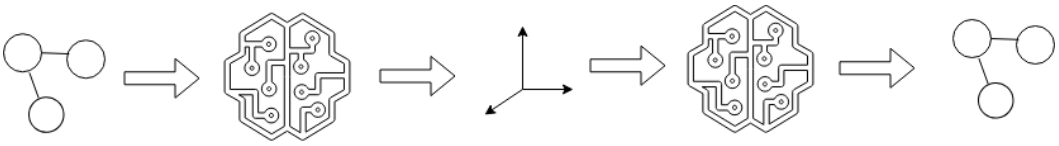


Figure 2.12: Illustration of KG completion with embedding in vector space. Inspired by Harmelen and Teije [78]. The three connected circles represent an KG, the brain with the nodes are a symbol for an ML algorithm, while the graph with three arrows represent embedding in vector-space.

Ontology Alignment

The fundamental concept of the SW is, as mentioned in the previous chapter, to gather information from multiple sources and use them together to answer complex queries. The different sources, however, might use different identifiers for the same concepts. This problem is equally applicable to managing multiple ontologies in the close world assumed systems. The identifiers can, for instance, be connected with the predicate `sameAs`.

American-site.us/`Soccer` owl:`sameAs` British-site.uk/`Football`

After the alignment, one can utilise triples from both ontologies together, which increases the expressivity.

Ontology alignment, also called ontology matching, is the process of mapping corresponding concepts from different ontologies together. This can be accomplished by hand. However, the process can be laborious, time-consuming and error-prone. A solution is to apply ML for the matching task, like Doan et al. approach General Language Understanding Evaluation (GLUE) [33].

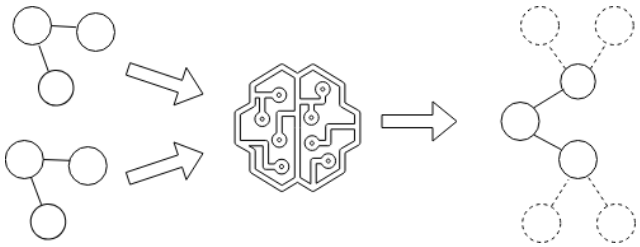


Figure 2.13: Illustration of ontology matching. As the last symbol shows, the outcome triples are connecting elements from both ontologies.

Explainable Artificial Intelligence

The traditional ML models can be viewed as black-box. The reasoning for the predictions are absent, and we are provided with the conclusions alone. Explainable Artificial Intelligence (XAI) is an academic umbrella term for transparent and explainable ML methods. However, the focus of this is usually on scientific analysis of the ML model, and not on providing an adequate explanation to the end-user. It has been argued that SWT can be a solution to enable human-centric explanations [68].

An example of how this could be accomplished in a CNN for image classification is to connect the hidden layers with the feature extraction to a KG [48]. One could, by doing so, end up with a descriptive explanation to how the system predicted the answer. For instance, claim it is a dog because it has paws and a muzzle.

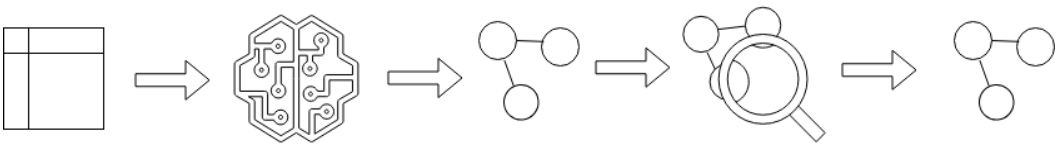


Figure 2.14: Example of an architecture where the reasoning can help explain predictions. Inspired by Harmelen and Teije [78]. The triples under the loop, represent the reasoning engine.

Reasoning and ML can be combined in multiple ways to provide explainable predictions. Figure 2.14 is just one architectural example among many.

Knowledge Graphs for Zero-Shot Classification

In Section 2.3.5, we introduced ZSC, the challenge of predicting unseen classes. We explained how the additional information connects the unseen classes with

the seen and enables us to predict the unseen classes as well. KG can be adopted to provide the system with this information. Figure 2.15 illustrates how we, with the use of embedding methods, can represent the triples based on their relations. When further using the embeddings in another neural network, the unseen classes can be evaluated based on their position in the vector space.

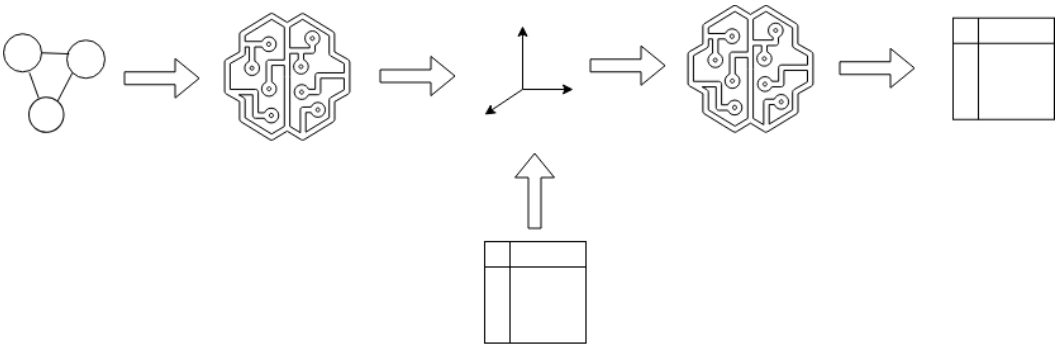


Figure 2.15: The simplified model describes how the embedding is learned from the KG, and later used in another ML algorithm when predicting from data. The data points receives a vector that describes them based on the KG landscape, and the ML model can learn the discriminating patterns.

There exist various ways that the two technologies can be combined, and this was only some examples. In our models, described in details in Section 4.3, we utilize embeddings in vector-space. The following section will, therefore, introduce the techniques of KG embedding.

2.4.1 Knowledge Graph Embeddings

To represent data in KGs have proven to be profoundly beneficial, as discussed in Section 2.2. However, it can be hard to manipulate and computationally expensive. Hence, the interests in KG embedding. The triples are embedded into a low-dimension continuous vector space, while the inherent structure is preserved [82]. Various embedding models exist, and the following paragraphs will present a few methods. Choosing a reasonable embedding model is essential in this project, both for performance, and ensuring to utilize the information in the KG fully.

Facts in the KG consist, as described in Section 2.2, of three items — subject, predicate, and objects. Embedding methods employ these as low dimensional vector representations of the triples. The individual research

uses different aliases for the same elements. Head, relation and tail (*i.e.*, h,r, and t) is in equivalent use. However, we will adhere to use subject, predicate and object, with the corresponding abbreviations S, P, and O.

Translating Embeddings

Bordes *et al.* proposed TransE [25], a translating embedding model described as "... a method which models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities."

Their objective is a model that is scalable, easy to train, and with a reduced number of parameters. In the embedding space, the translations represent the relationships. The object should be close to the subject, considering that a vector depends on the predicate. For all entity and relation, only one single low-dimension vector is learned [25].

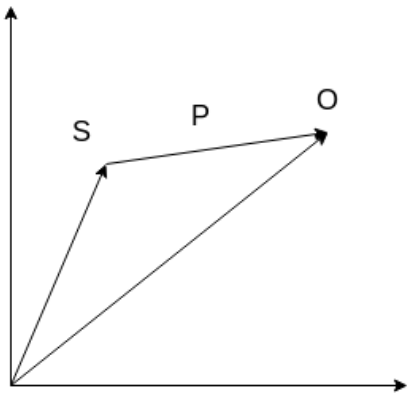


Figure 2.16: Demonstation of TransE.

The translation-based model works by minimizing the distance between the object and the subject plus the predicate (*i.e.*, S + P $\approx$ O). The scoring function can be implemented by moving the object (*i.e.*, O) over at the same side as the rest of the variables and end up with the equation S + P - O $\approx$ 0. Hence, true triples should get a score close to zero.

This is only true if the triple holds, if not, they should have reasonable distance and a score above 0. It is essential to consider this when optimizing to prevent all the entities from ending up at the origin (*i.e.*, give all the triples the position [0, 0, 0]). Random false triples can be added to the training set to prevent this. Hence, by giving the true triples output of 0, and the fabricated ones an output of 1, we can apply ML to learn a vector space representation

[25].

Hierarchical relationships are according to Bordes *et al.* widespread in KGs, and thus their primary motivation. Figure 2.17 shows how TransE expresses the hierarchy, similar to a tree-model. The siblings in the graph are adjusted the same height in the x-axis and grouped based on the parent. Along the y-axis, we get the parent to children relations.

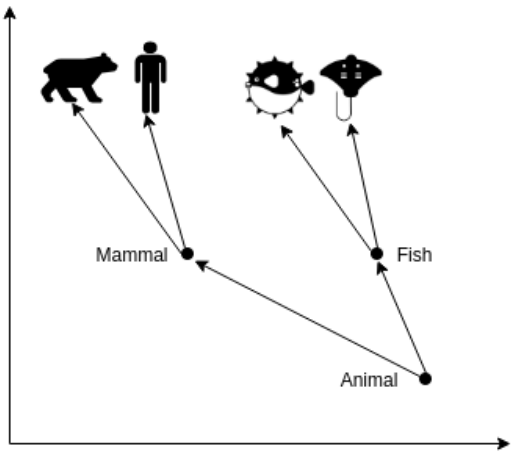


Figure 2.17: Hierarchy in TransE.

Holographic Embeddings

The objective of HolE [58] is to combine the expressiveness of tensor product with the simplicity of TransE. For this, circular correlation is used on the vectors to represent relations between entities. Tensor product (*e.g.*, RESCAL) can be explained as a neural network where a layer between the output and the input (*i.e.*, subjects and objects), contains as many nodes as the subject and objects multiplied ($\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^{n^2}$). The middle layer, when using circular correlation, has the same number of nodes as the number of subject-object pairs ($\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$). This reduces the memory and runtime complexity for compositional representation from quadratic ($\mathcal{O}(d^2)$) to linear ($\mathcal{O}(d)$) and linearithmic ($\mathcal{O}(n \log n)$), respectively. When looking at the memory complexity considering the embedding models it is the same for HolE as for TransE: $\mathcal{O}(n_e d + n_r d)$ [58].

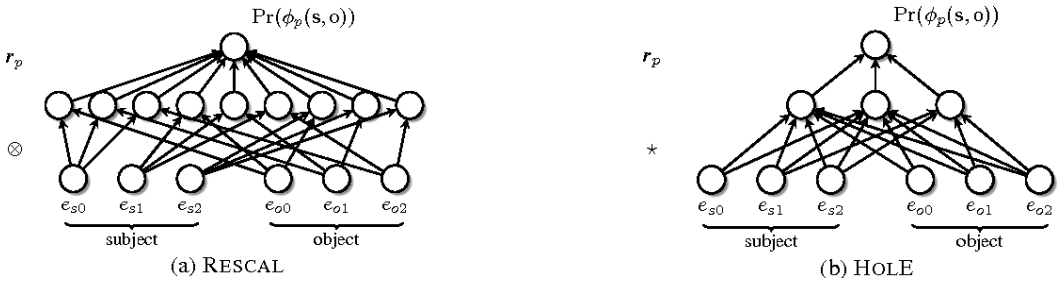


Figure 2.18: RESCAL and HolE as neural networks [58]

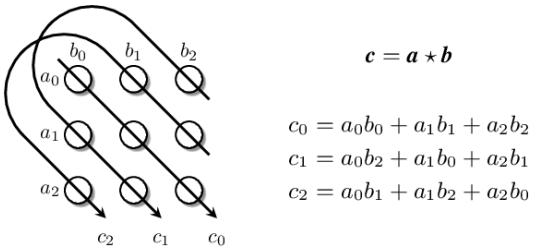


Figure 2.19: Circular correlation in HolE [58].

Bilinear-Diagonal Model

Yang et al. [87] introduce a general neural network framework for multi-relations representational learning. A two-dimensional matrix operator is used to represent the relations. By restricting it only to be a diagonal matrix, they achieve the same number of parameters as TransE. Results from their research show they manage to outperform not only TransE but additionally other more expensive models (*e.g.*, NTN).

In the research, Bilinear-diagonal and TransE are referred to as DistMult and DistAdd, respectively. As mentioned, both models share the number of parameters. The difference is the operand used on the two entity vectors. DistMult utilises a multiplicative operand (*i.e.*, element-wise dot product), while DistAdd utilises an additive operation (*i.e.*, element-wise subtraction with a bias). To avoid confusion, we will use the terms DistMult and TransE [87].

The Equivalence on HolE and Distmult

Hayashi and Shimbos prove in a study that HolE is equivalent to ComplEx [42]. At the same time, Trouillon *et al.* [75] explain that the ComplEx embedding is a version of DistMult, only in the complex plane. From this, we interpret that HolE as well is a version of DistMult, in the complex plane.

However, the operations in the complex plane enables the handling of anti-symmetric relationships.

Chapter 3
Related Work

In this chapter, we will present articles related to this master's thesis. The three central objectives are to get an understanding of the area of research, gain inspiration for our work, and introduce challenges we should consider. First, we explore research that combines Machine Learning and Semantic Web Technologies, as well as introducing a literature review for Explainable Artificial Intelligence. Then we further describe the Knowledge Graph Embedding for Ecotoxicological Effect Prediction, that, as specified in Section 2.1.1, are the main inspiration behind this work. The last two articles present two challenges we face during this master project. The first is concerning how sparse KGs perform poorly with knowledge graph embedding. The second article addresses the concerns for established embedding approaches (*e.g.*, TransE) not to consider the context of the triple properly.

3.1 Monitoring of Building Energy Consumption

The building sector was accountable for 40% of the energy consumption in the U.S by 2016. Both ML and KGs have been independently considered in several studies within the building domain to optimize this. Parastoo Delgoshaei *et al.* explores the opportunities for combining the two AI technologies to accommodate an enhanced intelligent building system [32].

The architecture of the system is divided into two parts, the Knowledge Representation component (KR-component), and the Machine Learning component (ML-component), which they implemented in Java and Python, respectively. For the two components to interact, they use a Python module

called JPype [52]. With JPype, it is possible to access Java libraries with Python.

There are three types of input to the system: Jena rules, ontology, and XML data files. The XML data files are sent to the ML-component and hold information on solar radiation, outdoor temperature, wind speed and KG data. The rules and the ontology is sent to the KR-component and facilitates the reasoner. The ML-component later uses the reasoner in its prediction.

The result shows that the prediction is close to the actual energy consumption, especially in the low energy consumption range. The research does, however, not compare to the result without using the semantics. Therefore, it is not clear to what extent the KR-component facilitated the ML-component. Their long-term vision is that this framework can be utilized for buildings-to-grid integration.

3.2 CORL for ZSC

In a research paper from the National University of Defence Technology [49], they propose a method they call CORL for ZSC. CORL is a abbreviation for Combining Ontology and Reinforcement Learning, and ZSC for Zero-Shot Classification.

The task is to predict animals based on their attributes. They have two types of input in their model, an ontology that hierarchically connects the animals, and a binary matrix where each animal have a value of either 0 or 1 for each attribute. The value 1 means that the animal has the traits, while 0 means that it does not. A chimpanzee has, for instance, the entries: swims = 0, bipedal = 1, hands = 1.

Reinforcement learning is used to arrange both the attribute annotations and the ontology entries by its discrimination for each animal. In the ZSC, only the most discriminating rules are used (*e.g.*, IF forest = 0 AND carnivore THEN Class = feline, where forest attribute comes from the training data, and the carnivore is parent of the feline in the ontology). They conclude that this approach achieves higher accuracy than baseline classifiers, while also highlighting the discriminating parts of the data.

3.3 Logic Tensor Network for Semantic Image Interpretation

Logic Tensor Networks (LTN) are a statistical relational learning framework that uses first-order fuzzy logic to learn from noisy data with logical constraints. In [34], Ivan *et al.* implement an LTN to classify image bounding boxes and detect relevant part-of relations between objects.

Logical operators are described using Likasiewicz t-norm2 [34, p. 5]. Take for instance the definition of the $\wedge$-operator:

$$G(\phi \wedge \psi) = max(0, G(\phi) + G(\psi) - 1)$$

The grounding G, of $\phi \wedge \psi$, is determined by adding the grounding of the operands and subtracting it by one. If the ϕ have the truth-value 0.8, and ψ have the value 0.9, the result will be 0.8 + 0.9 - 1 = 0.7. It holds less true than both the operands, which makes sense as adding a $\wedge$ operator decreases the possibilities of a true outcome since both have to be true. If the value is negative, the function will return 0, indicating an untrue statement. Having logical operators redefined with fuzzy logic, one can describe statements such that cats have tails. If the algorithm comes over a cat without a tail, it should not exclude it completely, even though it is more likely to be another animal [34, p. 4].

3.4 Knowledge Enhanced Neural Networks

The research conducted by Daniele and Serafini [30] proposes KENN (Knowledge Enhanced Neural Networks). They do so by adding a new layer that includes a set of learnable parameters they call clause weights.

The approach outperforms the state of the art methods on multi-label classification. The best results were on Zero-Shot Classification, where their method outperforms the runner-up by 10% [30]. Multi-label classification is the focus of this research, but they claim that the general framework and theoretical results hold for relational data as well. A secondary objective is giving information about the influence of each constraint. The clause weights are used to evaluate this.

3.5 Knowledge Graph Embedding with Entity Neighbors and Deep Memory Network

Kai Wang *et al.* [81] introduces a new model for KG embeddings, that provide additional information on the neighbourhood of the entities in the triples. The subjects and objects of the triples are passed trough a Deep Memory Network Encoder (DMN) before they are sent to a traditional embedding function (*e.g.*, TransE). The DMN integrates information from the entity neighbours to the embedding. The approach is shown to outperform the baseline TransE and other KG embedding methods, on graph completion tasks.

3.6 Semantic Web Technologies for Explainable Machine Learning Models: A Literature Review

This literature review discusses methods for using SWT to explain ML prediction. Research on Explainable Artificial Intelligence (XAI) has previously been explored, but the approaches commonly rely on technical analysis of the models. The review understands that "explainability is highly dependent on the usage of domain knowledge and not data analysis alone." from the authors Cherkassky and Dhar. Furthermore, they argue that "Semantic Web Technologies might be the key to achieve truly explainable AI-systems" [67].

XAI was researched comprehensively between 1970 and early 1990s. The field did decline the following decades, but with the rise of ML, it has recently found it way back. Although the fields overlap in some areas, a clear overview of the combination of SWT and ML is still missing, according to the article [67].

Mainly, there are two models where SWT are used for explainability. Unsupervised embedded task and supervised classification. They present different approaches for both models, but we found the supervised classification section to be of higher relevance to this project.

In regards to supervised learning, one method is to map neurons to classes of an ontology. This way, class expressions can act as explanations. The

neuron-class relation can also be learned, linking the neuron's weights the semantic grounded domain knowledge [67].

Another approach is to include hierarchy in the input — this way, the training can learn the patterns of the superclasses as well. This is more relevant to interpretability [67]. However, this can be relevant to improve the predictions as well.

3.7 Knowledge Graph Embedding for Ecotoxicological Effect Prediction

Myklebust *et al.* [55] apply, as mentioned in Section 2.1.1, KG embedding to ML models for ecotoxicology effect prediction and risk assessment. The intent is to predict the hazards of chemicals not yet tested in the laboratory.

Myklebust *et al.* use three models, where the first is the baseline model called "Nearest-neighbour". The model rests on current classification used by NIVA, and uses the most similar compound, defined by the hierarchy distance, to predict the effect on species. The second model is a multilayer perception, a neural network with hidden layers without using the KG. The last model, however, embeds the KGs with the ML model. It considers chemical similarity and shared attributes, as well as the hierarchy.

There was only a marginal improvement between the baseline model and random choice, making it an inadequate solution. The second model, the neural network without KG embedding, performs better than the baseline. However, they point out that random choice, when predicting unseen classes, will give a balance between positive and negative false prediction, resulting in good accuracy, while not producing interesting results. The models using the KG, however, is a more fitting approach to predict and provide interesting and unseen recommendations to the laboratory.

All the embedding methods seem to perform better than the model without embedding. They performed rather similarly, but some small variations separates them. In terms of recall, the best embedding methods are HolE at a decision threshold[1] of 0.5. At a decision threshold of 0.3, all the methods got a better recall. However, DistMult is the only one that maintains accuracy.

[1]Decision threshold sets the line of which output values to classified as true and false.

3.8 Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short

Pujara *et al.* [63] acknowledge the fact that current embedding techniques are evaluated based on the benchmark datasets (*e.g.*, Freebase and WordNet). The data sets are usually dense and reliable as humans have curated them with their field expertise. In their research, they performed empirical experiments to confirm the effects when the sparsity and errors increase in the KG. The researchers generated false triples to FB15k to analyze embedding methods sensitivity to unreliability, while removed triples to analyze the sensitivity to sparsity. The results confirm their theory, and we can see TransE and HolE both declining in HITS@10[2]

3.9 Knowledge Graph Embedding with Triple Context

Jun Shi *et al.* claims that current embedding approaches, including TransE, ignore relevant graph structure by viewing the triples separately. They introduce a Java-implemented embedding method called Triple Context Embedding (TCE) [70] as a solution to the problem.

TCE considers two specific context structures. The first is what they call the neighbour context. The purpose is to cover the local structure of the entity and its surrounding. The neighbour context is defined as $C_N(e) = \{r, t | \forall r, t, (e, r, t) \in K\}$. In other words, it only includes outgoing relations and the corresponding tail. We later in this project considers the context of the triples when giving them scores. Similarly to neighbour context, we use the term directly connected entities and entities in common, where we in some methods include incoming relations as well. We can define it with descriptive logic as $C_M(e) = \{r, t | \forall r, t, (e, r, t) \in K\} \cup \{h, r | \forall r, t, (h, r, e) \in K\}$.

Consider the following triples:

[2]HITS@K is an indicator of how many per cent of the true triples end up in the top-k elements. When removing a triple, the denominator will decrease and compensate for the lost opportunity for that triple being in the top 10.

Cats	Are	Animals
Dogs	Are	Animals
People	Like	Cats
People	Like	Dogs

$$C_N(Cats) = \{\langle Are, Animal\rangle\}$$

$$C_M(Cats) = \{\langle Are, Animal\rangle, \langle People, Like\rangle\}$$

The second context type is called path context and regards the relation-paths between a pair of entities. We equivalent use the term indirectly connected entities (see Table 6.4).

Even though TransE takes into account the parent-child relationship when embedding in vector-space (see Figure 2.17), Jun Shi *et al.* shows that it handles complex relations bad (*e.g.*, one-many, many-one and many-many). The relation categories many-to-one, and one-to-many when predicting subjects and objects, respectively, are still performing worse than the reversed types. DistMult, nonetheless, doubles the performance for the concerned relation categories, while still performing slightly better than TransE in all of the other categories, with only one exception[3], according to another study [87].

Jun Shi *et al.* only compares TCE to translating embeddings (e.g. TransE) and not to DistMult. We observe that both the research of DistMult and the research for TCE compares HITS@10 with TransE in the same relation categories. The experiments are on TCE performed on FB15k, while FB15k-401 (a subset with only frequent relations [86]) on DistMult. The results for TransE are substantially differing from the two tests. Despite this fact, it could look like TCE are out-performing DistMult in the challenging relation categories, but perform worse on the other categories. We cannot conclude from this analysis, other than acknowledging the possibility that TCE could be more beneficial than DistMult for some KGs.

3.10 On the Relevance to this Project

We will, in this section, justify why we included each of the related works, and how they are relevant for this master project.

[3]When predicting objects in many-to-one relations, the embedding methods TransE and DistMult receives the HITS@10 scores 83.2 and 81.0, respectively.

Monitoring of Building Energy Consumption

The main interest of this article is to see how they combine Ontologies with ML. As they chose to use two languages for their project, the architecture gets a clear division between the semantics component and ML component, with minimal interactions between them.

It is useful to know that libraries such as JPype exist, not to be bound by one programming language for both paradigms of AI. If we were to chose two programming languages, a divided architecture similar as they introduce could be reasonable to consider.

CORL for ZSC

The CORL for ZSC research has a similar challenge to what we face, with the same objectives. They combine ML with ontologies specifically for ZSC; however, a similar approach could have applied to our challenge. In addition to predict, they find the discriminating parts of the data and opens up the black box. Although their data differs from ours, we can draw inspiration form this in our project.

Logic Tensor Network for Semantic Image Interpretation

This approach applies first-order fuzzy logic in order to predict images. It is interesting to see, for inspiration purposes, how fuzzy logic can be used to both prioritize the semantics and improve the prediction. However, This is a project more directed to the trust of the KG, rather than finding discriminating facts. It also requires modelling specific to the logical constraints like "and" and "or". Our graph only contains subclass, type and domain-specific relations, which could be challenging to model in a similar manner.

Knowledge Graph Embedding with Entity Neighbors and Deep Memory Network

This piece is included to highlight a method of strengthening the context of triples in their embedding. The strategy has shown to outperform the baseline method of using TransE directly for the KG completion task.

Knowledge Enhanced Neural Networks

The Knowledge Enhanced Neural Networks (KENN) approach works by adding a new layer to the neural network in order to connect to the KG to the process. We included this article mainly for the same reasons as previously mentioned ones, to get inspiration and explore already existing approaches for similar challenges.

Semantic Web Technologies for Explainable Machine Learning Models: A Literature Review

The secondary objective of this project is to open up the black box by finding the discriminating triples, and conceivably be able to learn from this. Thus, it was of interest to read a literature review on XAI. This was to give a brief insight into what has been done in this area of research before.

Knowledge Graph Embedding for Ecotoxicological Effect Prediction

Knowledge Graph Embedding for Ecotoxicological Effect Prediction is the main inspiration behind this work and is therefore included in this chapter.

Sparsity and Noice: Where Knowledge Graph Embeddings Fall Short

This article highlights a critical shortcoming with KG embeddings. When the sparsity of the graph increases, the performance decline rapidly. This factor is taken into consideration when developing our approach and therefore included in this chapter.

The paper uses KGs derived from text and images as an example of sparse and noisy sources. The main source of the data in this work is the Toxicological Effect and Risk Assessment (TERA) KG [54]. As shown in [54], the KG has relatively little noise, although it is very sparse. TERA provides all relevant information on the chemicals. Therefore, the KG also includes non-discriminating triples.

Knowledge Graph Embedding with Triple Context

The embedding method Triple Context considers the context of the triple, claimed to be ignored by the established embedding algorithms. As we later

introduce in Section 6, our approach also takes the context of the triple into account, and it is essential for us that the embedding is capable of using that information.

Chapter 4
Framework

In this chapter, we first consider various programming languages and ML frameworks. The alternatives are numerous, and all can not be explored, but we present the few we have acknowledged. After justifying our choice, we describe the ML models we use in this project.

4.1 Programming Language

When choosing the programming language for this project, we considered two in particular; Java and Python. The reasons for this are as follows. We have prior experience with them both. The languages are widely used and with proper documentation available. Lastly, useful libraries are accessible to abstract accepted functionality with well-established frameworks.

Java was in 1991 developed by Sun Microsystems, who believed that the union of digital consumer devices and computers was the next big wave in computing [9]. Oracle acquired Sun in 2010, and have continued to maintain, and further develop the language [18]. Java is strong typed, object-oriented, and compiles down to bytecode instruction, making it platform-independent [14].

Working with SWT in Java, the Apache framework Jena [15] provides libraries to handle RDF, RDFS, OWL and SPARQL. New facts can be explored by reasoning with a rule-based inference engine. We will, however, not reason in this project, but the framework also provides several strategies to store the RDF triples [20]. It also exists ML frameworks for Java. Deeplearning4j [12], Apache Spark [7] and Weka [4] are all Java libraries for AI. Tensorflow [19] is also available for Java but not as fully developed as in Python.

Guido van Rossum developed Python in the early 1990s at the Stichting Mathematisch Centrum [6]. The language is interpreted, meaning it executes instructions directly without compiling the source code on beforehand. It is also dynamic typing and binding, in contrast to Java. Python is, like Java, object-oriented [17]. Python is a more abstracted language, resulting in fewer code lines. According to a survey from Developer Economics, 57% of 2'000 data scientists and ML developers use Python as their programming language, making it the winner of the study. Java arrives on the third place, coming right behind C/C++, with 41% using it [79].

Our choice for programming language lands on Python. Python is powerful for solving ML problems and contains as discussed convenient frameworks. We would also argue that Python has more than enough RDF handling for this project (*e.g.*, RDFLibs [24] triple store and SPARQL [24]). Other imported projects (*e.g.*, TERA [54]) use Python, which furthermore makes it convenient for us to use.

4.2 Machine Learning Framework

There is no shortage of ML frameworks to choose from in Python. Similar to the programming language, we can not consider all, but we will discuss a few well-used libraries.

Facebooks AI research group have since 2016 developed PyTorch [60]. PyTorch implements dynamic computation graphs, enabling network behaviour to change without starting from scratch [57].

Theano [74] was developed back in 2017 and is now maintained by MILA (Montreal Institute for Learning Algorithms). This is a cross-platform project suitable for DL. However, the project is no longer under development. It can also be challenging to use without wrappers since this is a lower-level Application Programming Interface (API) [57].

Another popular tool is Scikit-Learn [61], which is strongly linked with statistic and scientific Python packages. The Anaconda distribution of Python does, therefore, includes Scikit-Learn. Scikit-Learn is however not comprehensive enough [57], and might not be suitable for our project. It comes with several premade models, although it has limitations when it comes to customization possibilities.

In November 2015, Google Brain released the open-source framework TensorFlow [21] under the Apache 2.0 licence. TensorFlow is the most popular framework and is evolving fast [57]. However, this is a lower-level API and can be challenging to use alone. Several other tools build on top of TensorFlow to resolve this issue.

Francois Chollet's Keras [27] is one of those tools. The wrapper library was released in July 2018 and utilized other frameworks, among them TensorFlow, making the set-up clean and convenient. Using this framework will allow us to focus on defining the model and abstract the backend. This advantage consequently introduces a weak spot; the tool becomes less flexible [57]. Today Keras is a part of TensorFlow and provided as a submodule [16].

It looks like the Keras framework is comprehensive enough for our ML models, which is presented in Section 4.3. The `Keras.layers` modules provide the classes `Embedding`, `Dense` and `Dropout`. They allow us to convert chemical id scalars into vectors, create hidden layers and apply dropout, respectively. Keras also allows multiple outputs. This functionality makes it easier for us to implement our model.

4.3 Machine Learning Models

This system is composed of various components that we will outline in this section.

As we can see in Figure 4.1, the system takes in a scored KG. By having a score corresponding to each triple, we allow the model to prioritize them and address weights for the triples. We will deliberate in details on prioritizing triples in Chapter 6. The baseline method only distinguishes between true and false triples, where the triples get score 1 and 0, respectively.

Before entering the models, the two inputs are balanced to the same length (explained in Section 7.1.2). In this case, the KG is considerably more massive, and the training data is randomly duplicated to align. This means the size of the KG can affect the results.

We use two Keras models. The first takes in a KG in addition to the training data, therefore, called KG-model. It uses the embedding algorithms to utilize the information in the KG to benefit the prediction. The second is a One Hot Model, abbreviated as ON-model. This does not use any additional

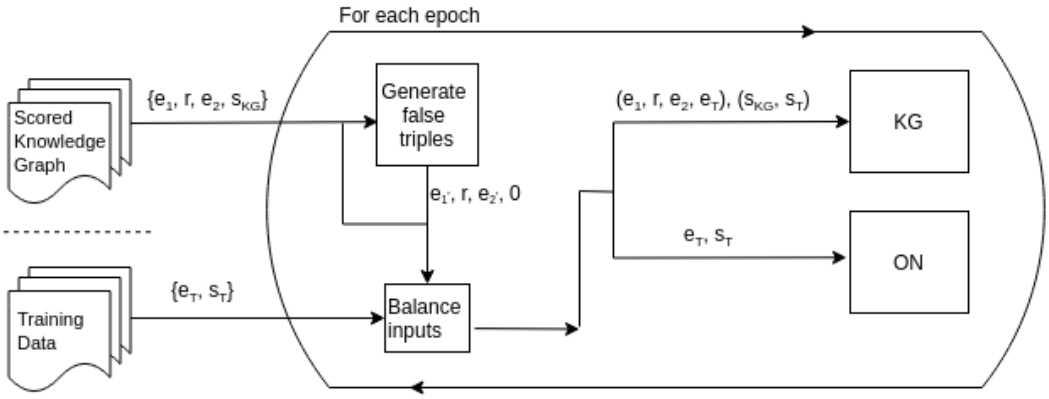


Figure 4.1: This is an abstraction of the elements in an epoch. e_1, r, and e_2 is the vector representation, while e_T is used for the chemical in the training data. They are all subscripts from the variable e, to indicate that they are from the same set, except for r which can not be a chemical, but only a predicate. As for the scores (S), we have separated them with the subscripts KG and T to indicate that it is the score of the triples and the chemicals in the training data, respectively.

knowledge. Considering that the test data is not connected to the training data in any means, it should perform close to random. The ON-models are, however, trained under the same circumstances as the associated KG-models and used as a baseline. This is to give a better comparison between the KG-models, as the conditions can vary from each approach (*e.g.*, size of the KG).

Figure 4.2 illustrates the KG-model in more detail. All the entities and relations are trough embedding layers converted from scalar ids to vectors, creating matrices from the input vectors. The corresponding triples from the KG are sent into the embedding algorithm before returning as the first output.

The corresponding entity from the training data is sent through a neural network consisting of two hidden layers with 16 neurons and a dropout rate of 0.2. The output layer consists of only one single node to determine if the compound is toxic or not. The value is returned as the second output in the models.

4.3.1 Knowlegde Graph Embedding

Training the neural networks is time-consuming. The number of runs is the product of the quantity of chosen embedding algorithms, the proposed solu-

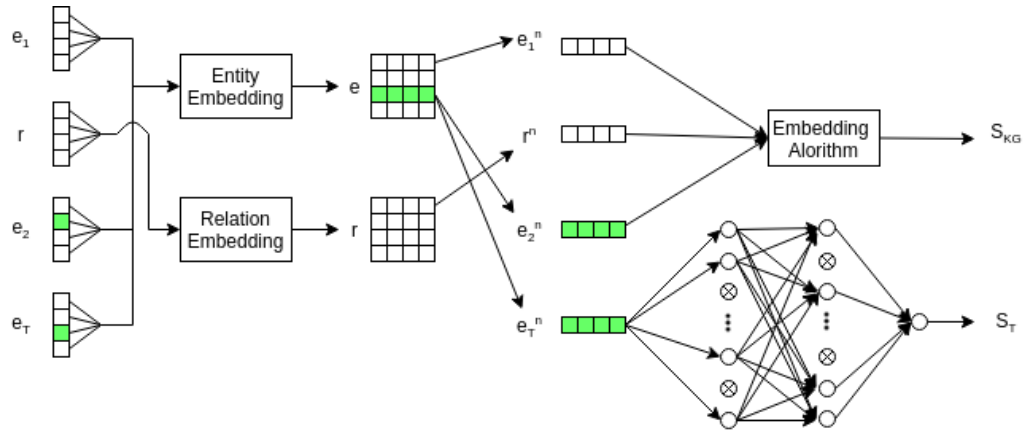


Figure 4.2: The KG model in more detail. As in Figure 4.1, e_1, r, and e_2 is the vector representation, while e_T is the chemical in the training data. The outputs of the model are S_{KG} and S_T, which represents the KG embedding score, and the score from the training data, respectively. The blocks coloured green represent the same chemical to illustrate how they are connected.

tions, and repeated simulations. ML models are not deterministic, and the execution must be measured repeatedly to get a representative average. Hence, narrowing down the number of embedding methods and their performance is essential in this project.

In Section 2.4.1 we point out that HolE is a version of DistMult, only in the complex plane. Despite HolE handling antisymmetry and being a well-established embedding algorithm, we would argue that it is sufficient to use DistMult and TransE for the scope of this project. They are both efficient in terms of complexity compared to the other discussed methods. DistMult have as mention outperformed TransE and other more expensive models [46], while TransE still considers directed graphs.

Under relevant work, Section 3.9, we introduced an embedding approach (TCE) that priorities the context of the triple, and claims TransE of only considering triples separately. TCE has proven to outperform TransE. However, we choose not to include it in the scope of this project. It is implemented in Java and would have required reimplementation. It is also worth repeating that keeping the number of embeddings algorithms in our project to a minimum is of great benefit to us because of the increasing runtime it demands.

Thus, we would argue that the two embedding algorithms, TransE and DistMult, is enough for proof of concept in the scope of this master thesis.

```python
import tensorflow as tf
from keras.layers import Lambda, Activation

def TransE(s, p, o):
    l = Lambda(lambda x: 1 / tf.norm(x[0] + x[1] - x[2], axis=-1))
    score = l([s, p, o])
    score = Activation('tanh', name='score')(score)
    return score
```

Figure 4.3: Python implementation of TransE.

Figure 4.3 shows how we implemented TransE in Python. All the triples are first sent through a lambda function. The function does, as TransE implies, subtract the object value from the sum of the subject and the predicate values, in order to generate a score. The function tf.norm is applied to find the Euclidean distance, *e.g.*, the straight lines in the tensor ($|s + p - o|$).

As we know from introducing TransE, a low score, preferably close to zero, indicates that the triple is true. However, we want to interpret the outputs from the function the other way around. Hence, the values close to one as true, and the values close to zero as false. Therefore, we need to convert the score from TransE before sending the scores through the Tanh function. In order to give a high score a low value, and a low score a high value, the value can be applied as the divisor for the number one. For instance, the equation $1/100$ equals 0.01, while the equation $1/0.01$ equals 100.

```python
from keras import backend as K
from keras.layers import Lambda, Activation

def DistMult(s, p, o):
    l = Lambda(lambda x: K.sum(x[0] * x[1] * x[2], axis=-1))
    score = l([s, p, o])
    score = Activation('sigmoid', name='score')(score)
    return score
```

Figure 4.4: Python implementation of DistMult.

Figure 4.4 shows likewise how we implemented DistMult. This approach is similar to what we did with TransE, but with another lambda function. The function K.sum(s * p * o) is a way to calculate the "Hadamard product", *i.e.*, the element-wise dot product ($S \circ P \circ O$). This leaves us with a matrix of batch size times 1 (*i.e.*, a vector), were the corresponding result is the score of the triple.

Chapter 5
Ecotoxicology Effect Data

This chapter will introduce the sources for the data we utilize in this project. We will describe the content and structure of the datasets. In addition, we provide an analysis for the sparsity in the Knowledge Graph.

A limiting factor in risk assessment for ecotoxicological research is the access to data for chemical effect on species. Testing this can require up to hundreds of organisms, that is both laborious and can raise ethical questions for the welfare of the animals (see Section 9.7). Hence, it is beneficial to collect data from public sources. The data structure varies from source to source and can be in the form of tabular, RDF and SPARQL endpoints [54]. The heterogeneous sources need to be on the same format for us to use them in our model. SWT is a solution to this issue. In Section 2.2, we describe how it is achievable to express nearly all information with only three components. Storing information in a graph-based database is an excellent solution when combining various sources.

The Toxicological Effect and Risk Assessment Knowledge Graph, abbreviated as TERA [54], is such a solution. As mentioned, ecotoxicology is a broad topic that involves various research fields. Hence, effect prediction depends on different types of data to assess the outcome of unseen pairs of chemicals and species. TERA collects not only effect data but also compound data, taxonomy and species traits. To limit the scope of this master thesis, we only focus on one species (fathead minnow), and thus only collect the compound data. However, our methods can easily be extended to include more species. In the following sections, we will highlight which parts of TERA we used.

Effect Data

The effect data specifies which chemical that is toxic for which species. In 1996 a database titled ECOTOX[1] [5] was released for governmental users in the U.S. Four years later it was released to the public with a web-based interface. The system integrates data from three independent databases: AQUIRE with aquatic life data, PHYTOTOX with terrestrial plants, and lastly terrestrial wildlife from TERRETOX.

result_id	test_id	endpoint	conc1_mean	conc1_unit
2063723	2037887	LC10	220	mg/L
...	...	...	...	...

Table 5.1: Ecotoxicological effect data

The data is originally tabular and contains endpoints, concentration means, and the concentration units. However, it is arranged by TERA into triples. In Section 2.1, we describe the different endpoints like LC50 and LD50. The example above uses LC10 and expresses that 10% of the population will die when it is exposed to 220 mg of the chemical per litre of water. When analysing, it is necessary to compare rows with the same endpoint and unit. It is feasible to convert some units to make them relative. It is important to be aware of the several factors that can affect the results. The temperature, pH-value and ionic content in the water, the duration of the exposure, and organism traits like age and size can all influence the outcome [54].

Other tables in the ECOTOX database contains additional information regarding the compound and species. However, the information is limiting, and external sources should complement the data [54].

Compounds

The compound dataset holds information on the chemicals, both the hierarchy structure and their characteristics. TERA uses PubChem[2] as a base. Pubchem is an open (*i.e.*, the users can themselves enter their research) graph database released in 2014 [64]. We can use the data directly as it already is

[1]ECOTOX data download: https://cfpub.epa.gov/ecotox/
[2]PupChem data download: https://pubchemdocs.ncbi.nlm.nih.gov/downloads

stored in triples. TERA further supplies a dataset called ChEMBL to compliment with the hierarchy. The relevant triples are retrieved by querying their SPARQL endpoint ChEBI[3]. The query searches for items that have an edge to one of the relevant chemicals [54].

The Medical Subject Headings, abbreviated as MeSH[4], is a vocabulary created by the National Library of Medicine. Specialists continuously expand and alter the database, with thousand annually updates [10].

There are three basic types of MeSH records: Descriptors, Qualifiers and Supplementary Concept Records (SCRs). Descriptors are used for indexing and retrieval of articles. It additionally describes what the subject is about, its genre, and geographical information. Qualifiers are used to supplement the Descriptors, and group together records that have the same aspect of the subject (*e.g.*, to indicate that an article is about drug effect on the liver and not a general article about the liver). SCRs are used to index chemicals and drugs. These records are not arranged in a tree hierarchy. However, links to Descriptors connects the chemicals in such a matter [11].

There are other types in addition to the once mentioned above. The graph includes predicates like "see also", "term" and "pharmacological action". The last-mentioned is considering how the chemical or drug behaves in the body or the environment (e.g. pharmacological action insecticide) [8].

Subject	Predicate	Object
Veterinary Medicine	See Also	Laboratory Animal Science
Osteogenesis	Term	Bone Formation
Thymol	Pharmacological Action	Antifungal Agents

Table 5.2: Examples of triples with the predicates see also, term and pharmacological action.

Mapping to CID

The sources use different ID-types. Thus, to recognize the equivalent objects from the various systems as the same, we need to map to a standard ID. We have chosen to use the Compound Identifier (CID) (*e.g.*, from the InChIKey DDBREPKUVSBGFI-UHFFFAOYSA-N to CID4763). We use the Python

[3]ChEMBL SPARQL endpoint: https://www.ebi.ac.uk/rdf/services/sparql
[4]MeSH SPARQL endpoint: https://id.nlm.nih.gov/mesh/query

library PubChemPy [73], a wrapper for the PubChem API, to translate to CID. Note that not all IDs can translate (*e.g.*, MeSH IDs that points to concepts and not compounds). We will refer to the instances mapped to CID and triples that have such an object or subject, as CID-mapped.

5.1 Knowledge Graph Analysis

First, we tried only using hierarchy data from ChEMBL dataset in our experiments (presented in Part II). The graph gave a bad performance for the baseline method, and we could not see any significant difference between the KG-model and the ON-model. After analyzing the graph, we found out that it was not as connected as we initially assumed.

There are two types of relations in the KG: *has parent* and *type*. The *type* attribute is referring to other vocabularies and does not have CID-mapped objects[5] (*i.e.*, the triples with the predicate *type* can not directly connect two chemicals from the training data). However, two subjects can point to the same class and be connected in that matter. *e.g.*, Ethylbenzene and Styrene in Pubchem refer to the same class in the Bioontology vocabulary. This is unfortunately only true for four classes:

- http://www.biopax.org/release/biopax-level3.owl#SmallMolecule
- http://purl.bioontology.org/ontology/NDFRT/N0000008130
- http://purl.bioontology.org/ontology/NDFRT/N0000006718
- http://purl.bioontology.org/ontology/NDFRT/N0000179021

It appears that every subject relates to SmallMolecule, which is not a chemical identifier. The other *type* relations only have two subjects relating to them. We assume that these exceptions have a small impact when it only regards a few triples. Therefore, the *type* relation is excluded in the following set evaluation[6].

$$Leafs : Objects \setminus Subjects$$

[5]CID is the standard ID-type we try to map all the entities to.

[6]The sign $\setminus$ represent set subtraction (*i.e.*, elements in Objects, not present in Subjects).

$$n(Leafs) = 675$$

$$Roots : Subjects \setminus Objects$$

$$n(Roots) = 25157$$

$$Others : Objects \sqcap Subjects$$

$$n(Others) = 0$$

$$Objects \sqcap Subjects = \emptyset$$

$$Objects = Leafs$$

$$Subjects = Roots$$

Since no chemicals are both subject and object, the two sets will further be handled as two separate disjoint sets. The depth of the tree is consequently of length 2.

When excluding the type relation, there are duplicates among the objects but none among subjects. All subjects are therefore unique. Hence, objects can not relate to other objects in the KG. However, subjects can relate to other subjects by having an object in common.

Since the two sets of subjects and objects are disjoint, it was interesting to see if they are both included in the test and training data. It shows that most of the objects (662 of 675) are contained in the test and training data, although the training data only includes nine compounds from the subject set — while test data has none in common. Figure 5.1 illustrates the parts of the KG.

We could, indeed, switch training data with the test data, but since it is so sparse, we could assume it would perform poorly in any manner.

To extend the KG, we included the MeSH dataset. The MeSH graph is nearly five times larger than the ChEMBL graph. We observe after including MeSH that 29,348 nodes are internal nodes (*e.g.*, not leaves or roots), which means that the depth of the graph is greater than 2. Hence the objects and subjects cannot be treated as disjoint sets, as some of the nodes consequently are objects and subject at the same time. Therefore, an identical analyze is

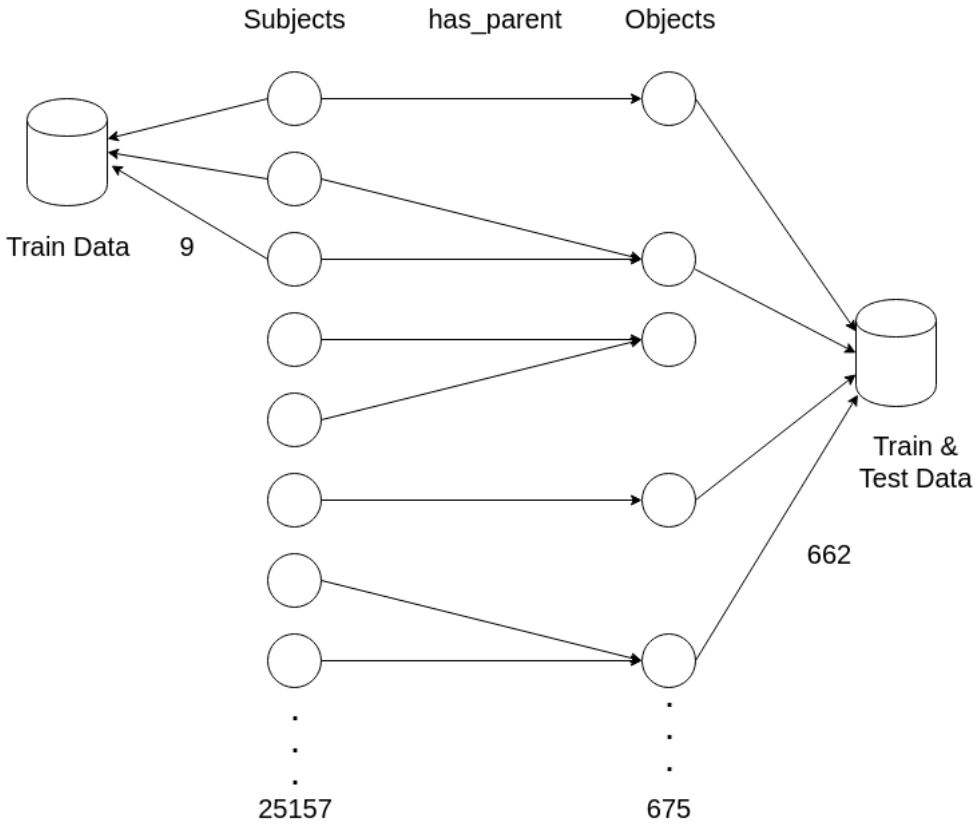


Figure 5.1: Structure when only using ChEBI.

not obtainable. However, we do observe that the graph is more connected. The KG has the same number of connection (662) from the objects to train and test data, but the subjects, have gone from 9 connections to 215. In addition, the duplicates in the subjects have now gone from 0 to 211, and the objects from 605 to 714. We have illustrated this in Figure 5.2, but keep in mind that the subjects and objects are not disjoint and cannot be interpreted equally as Figure 5.1.

After running experiments with the new KG, we saw the KG-model outperformed the ON-model. This indicates that the KG now has sufficient information to contribute to the ML model.

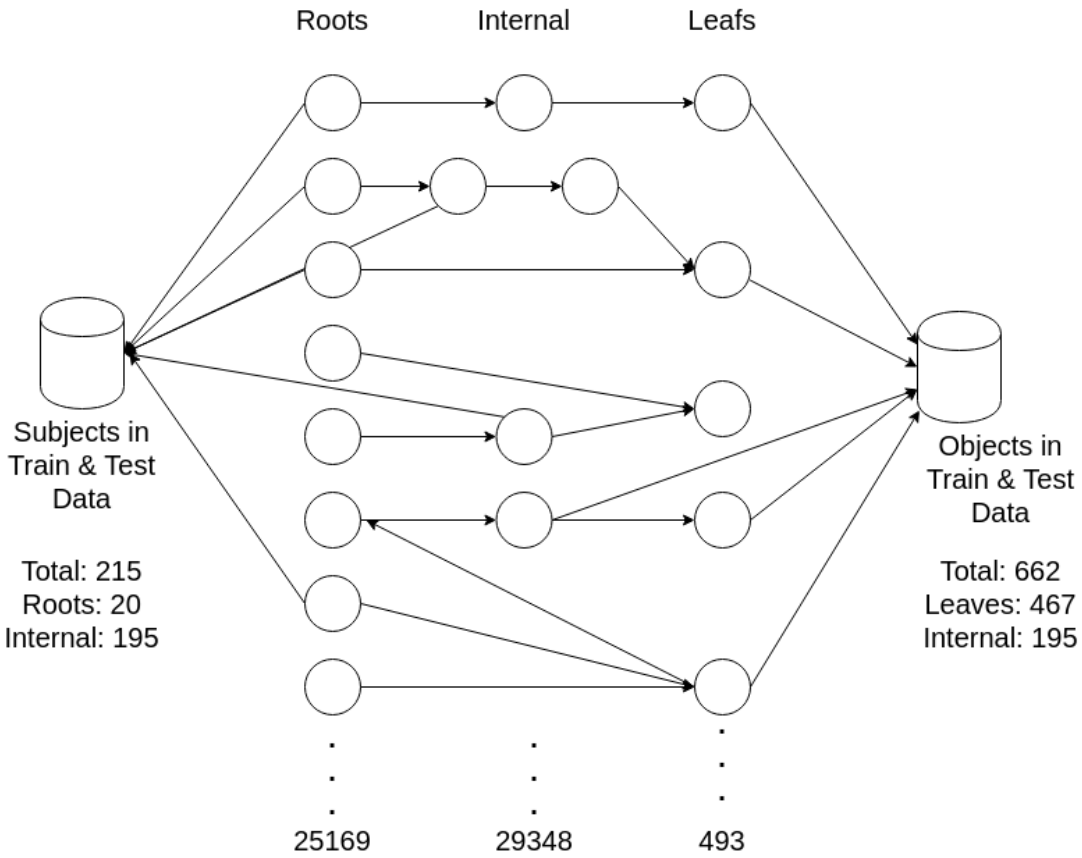


Figure 5.2: Structure when including MeSH. The connections with the internal nodes are for demonstration purposes and are not based on concrete examples.

Part II
The Project

Chapter 6
Proposed Approach

In this chapter, we introduce our proposed approach to improve and provide explainability to the predictions. The intention is to evaluate the triples level of discrimination according to their relations with toxic and non-toxic chemicals. First, we present naive approaches and the concepts behind central crawling and scoring algorithms. We later discover that the graph is too connected for our initial crawl and therefore giving us the same results as the trivial solution. To bypass this problem, we introduce various ways to limit the crawl to only concern the relevant triples.

Data, crawling methods and ML models are available at:
https://github.com/NIVA-Knowledge-Graph/mpnp

The Knowledge Graph (KG) is extensive and consists of over 155,000 triples. Inevitably, a significant amount of those triples are irrelevant to the prediction. As we observed in Section 3.8, KG embeddings fall short when using sparse and noisy graphs. Hence, filtering out unnecessary triples is believed to increase the precision of the prediction.

In the intention of filter out the uninteresting information in the KG, we introduce our hypothesis. We believe that discriminating triples are either connected to toxic or non-toxic chemicals. By scoring the triples, based on their connection to the chemicals, we aim to narrow down and prioritize the KG to better the prediction.

A secondary objective of this master's thesis project is to provide explainability for the prediction. Although this strategy is not explaining the individual prediction, it can provide us with information on the specific triples in the graph.

6.1 Naive Approaches

The first approaches practice a strategy of filtering the KG to exclude irrelevant triples, without dropping too many discriminating triples.

6.1.1 Only CID-Mapped (OCM)

We described in Chapter 5 how the different sources use different identifiers for the same chemical, and how we decided to use the PubChem Compound Identifier, abbreviated as CID. The other subjects and objects in the KG were translated to this ID, where it was possible. We define entities, and triples containing entities, that are mapped to this type of ID, as CID-mapped.

Subject	Predicate	Object
CID22970781	has_parent	CID1183

Table 6.1: Example from the ChEMBL dataset. Both **CID**22970781 and **CID**1183 qualifies the triple as a CID-mapped.

Subject	Predicate	Object
CID9237	pharmacologicalAction	Radiation-Protective Agents
M0006679	term	T012986

Table 6.2: Example from the MeSH dataset. The subject **CID**9237 qualifies the first row as a CID-mapped triple. The second row, however, has neither the subject nor the object mapped to CID and does not qualify as a CID-mapped triple.

Chemical	Concentration
CID4763	2.681

Table 6.3: Example from the training data. The row qualifies as a CID-mapped because of the chemical **CID**4763.

All the instances in the ChEMBL dataset, the training and the test data are CID-mapped. However, the triples from the MesH dataset have only 3.95% of the triples CID-mapped. Therefore, the remaining triples in MeSH cannot be directly connected to the training data. However, they are believed to complement the hierarchy and can play an essential part in the prediction.

We mentioned in Section 5.1 how the ChEMBL dataset was too sparse for us to use, which is the reason we added the MeSH data. However, it would be of interest to see how the models perform with only the CID-mapped MeSH triples in addition to the ChEMBL data. Hence we include this approach and assign it the name Only CID-Mapped, as it only includes CID-mapped triples.

6.1.2 Crawling the Graph

To our understanding, a substance from the test data needs to be connected to a substance from the train data in the graph for the graph to provide helpful information. It can be connected directly, indirectly, with other entities in common (*i.e.*, shared attributes), or a mix of the above. Triples that cannot be connected to any chemical in the training data are, consequently of no use in the training process. Table 6.4 shows examples of the different ways the triples connect. The arrow $\rightarrow$ is used to demonstrate predicates relating the entities.

Directly:	$A \rightarrow B$
Indirectly:	$A \rightarrow x \rightarrow B$
Entities in common	$A \rightarrow x, B \rightarrow x$
Mix	
- Indirectly:	$A \rightarrow x, x \rightarrow y$
- Entity (subject) in common:	$z \rightarrow y, z \rightarrow B$

Table 6.4: Examples of how the triples can be connected.

The basic idea of this naive approach is to crawl through the KG, starting with triples that are directly relevant for the training. At the end of the crawl, only visited triples are kept. The remaining triples are considered irrelevant for the learning process.

To use this approach, we need to find the relevant triples. First, we tried to start with all the triples with either CID-mapped subject or object. The benefit is that the new KG is independent of the training set. Should the model later be trained with new inputs, the same KG can be used. However, the resulting KG from this was identical to the original knowledge graph. In other words, all the triples connect to a CID-mapped entity.

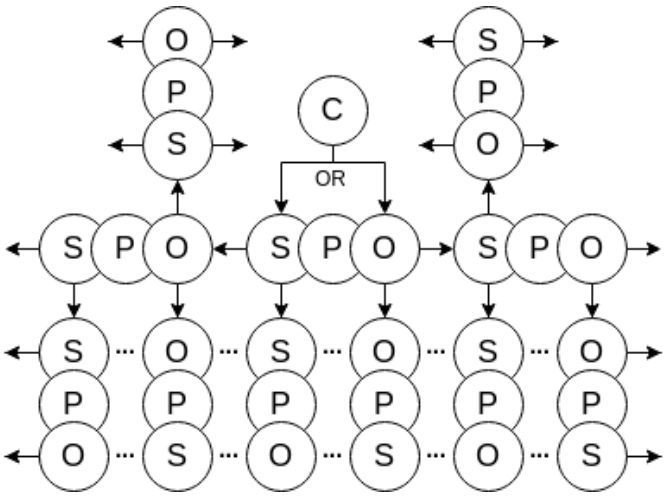


Figure 6.1: Examples of crawl from one chemical.

We also attempted to only start with triples that have subjects or objects contained in the training data. This would mean that it is necessary create a new KG every time the model is trained with new training data. The time used to crawl the graph is relatively similar to the training time. Adding this process as an initial process to the model would indeed increase the run time, depending on the size of the original KG and the training set. However, if this is proven to improve the prediction, the trade-off can be worth it. This Machine Learning (ML) training process is not continuous, and we assume only periodic training. The results are inadequately similar to the previous, with 94% of the KG kept. We could as assumed not see any improvements from only removing 6%.

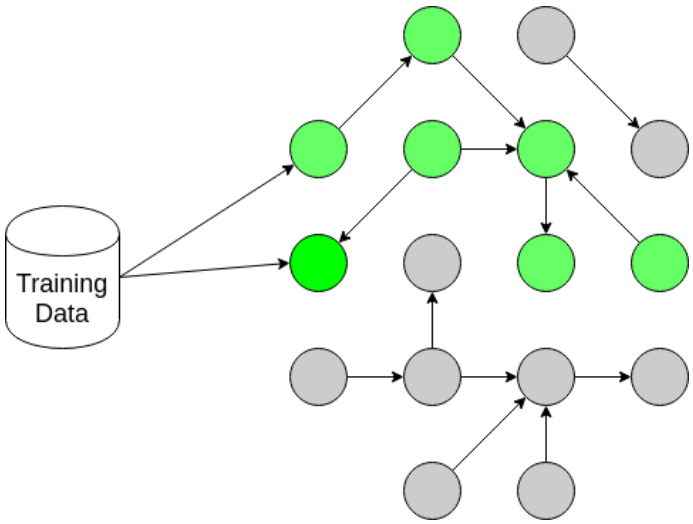


Figure 6.2: Example of the crawl when using chemicals in training data as starting point. The kept triples have green heads and tails, whilst the others have gray.

```
def crawl(kg, to_be_explored):
    global explored
    if not to_be_explored:
        return
    explored = explored | to_be_explored

    neighbors_o = [(s, p, o) for s, p, o in kg if o in to_be_explored]
    s = [s for s, p, o in neighbors_o]
    neighbors_o += crawl(kg, set(s) - explored)

    neighbors_s = [(s, p, o) for s, p, o in kg if s in to_be_explored]
    o = [o for s, p, o in neighbors_s ]
    neighbors_s += crawl(kg, set(o) - explored)

    return neighbors_o + neighbors_s
```

Figure 6.3: Implementation of the initial crawl algorithm.

6.2 Approaches with Scoring Triples

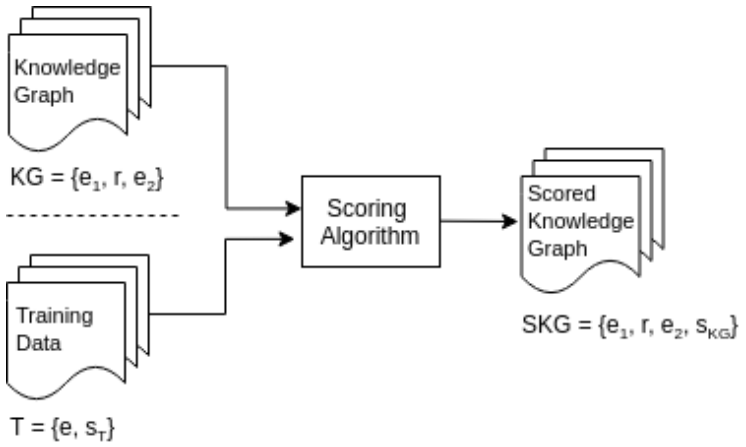


Figure 6.4: The scoring algorithm takes in the KG and the training data to return a scored KG.

In contrast to a lot of the relative works, this problem only have two labels: 1 (harmful) and 0 (not harmful). However, the original LC50 concentrations can say something about the degree of harmfulness. The lower the concentration, the more harmful a substance is. This method builds on the idea that triples related to chemicals that causes great harm, can be more discriminating.

To illustrate this, we present a particular example form the data. The entity in the MeSH dataset, D007306, is the URI for insecticides. The subset of chemicals with relation to this entity is designed to be poisonous for specific organisms. We discovered that the chemicals classified as insecticides have a relatively high concentration, with an average of 5.573 mg/L. This is compared to the other chemicals less toxic, as the average of all the chemicals is 4.042 mg/L. The reason can be that the insecticides target insects and that a much higher concentration is necessary for other species, such as the fish used in these tests. We will, however, use the example for demonstration purposes.

In the KG, the two compounds Phosalone and O-Ethyl O-(4-nitrophenyl) phenylphosphonothioate are connected as described in Table 6.5.

Subject	Predicate	Object
Phosalone	pharmacologicalAction	Insecticides
O-Ethyl O-(4-nitrophenyl) phenylphosphonothioate	pharmacologicalAction	Insecticides

Table 6.5: The table shows how Phosalone is connected to O-Ethyl O-(4-nitrophenyl) phenylphosphonothioate in the graph.

O-Ethyl O-(4-nitrophenyl) phenylphosphonothioate have in the training set a concentration of 6.209 mg/L. Phosalone is among the test data and has similar to the phenylphosphonothioate, concentration of 6.238 mg/L.

The algorithm displayed in Figure 6.5, is the initial idea of how to score the triples. The concentrations of the chemicals are subtracted by the median ($\approx$4) to divide the toxic and non-toxic chemicals as negative and positive numbers, respectively. All the triples start with a score of 0, which increases or decreases as the chemicals visit them with their crawl. In the end, the scores the triples retrieved are converted to positive numbers, to consider the triples discriminating toward toxic and non-toxic equally.

Triples connected to multiple chemicals can gain more points. This can mean that the triple is relevant. If the triples receive both negative and positive scores, this would balance each other out, which can mean that the triple is in fact, irrelevant. If the triple is both connected to toxic and non-toxic chemicals, the information it holds has presumably little influence when it comes to determining the toxicity of chemicals.

```python
def basic_scoring_algorithm(training_data, kg):
    chemicals, scores = list(training_data['cid']),
                        list(training_data(['y'])

    # start with 0 touches and scores for all the chemicals
    kg_score, kg_touches = initialize_empty_dicts(len(kg))

    # subtract the median of all the scores
    median = np.median(scores)
    scores = scores - median

    # crawl the graph from the chemicals
    for chemical, score in zip(chemicals, scores):
        neighbors = crawl(kg, chemical)

        # add the score to all the connected triples
        for neighbor in neighbors:
            kg_score[neighbor] += score,
            kg_touches[neighbor] += 1

    # convert to absolute value of the score
    kg_score  = {k: {np.abs(v)} for k, v in kg_score.items()}

    # sort knowledge graph based on score
    kg_dict, kg_touches = sort_by_score(kg_dict, kg_touches)

    return kg_dict, kg_touches
```

Figure 6.5: The basic scoring algorithm, simplified. The term touches describe how many chemicals the triple is effected by.

6.2.1 Results from the Basic Scoring Algorithm

In Table 6.6, the first column "Rank" refers to the position of the triple based on the "Score", were 0 is the most discriminating. The column "Touched" refers to how many chemicals visited the triple and influenced its score.

Rank	Subject	Predicate	Object	Score	Touched
0	CID129760973	has_parent	CID289	71.440	456
1	CID129847363	has_parent	CID28780	71.440	456
2	CID67720914	has_parent	CID4101	71.440	456
		...			
146326	T690973	type	Term	71.440	456
146327	CID66608653	has_parent	CID10342051	10.592	2
		...			

Table 6.6: Results from the basic scoring algorithm.

The results from the basic scoring algorithm are not unexpected. The KG is profoundly connected, and 146326 out of the 155367 triples (94%) all have 456 "touches", and the same scoring. This method involves the same type of crawling as the previous method. We suspect that one or a couple triples, like X type SmallMolecule, connects almost all of the triples.

We could either remove this from the graph classifying it as trivial or introduce a general solution for this problem. When removing that triple, 140813 are still touched by the same 334 iterations. SmallMolecule was only the 97th most repeated object with 166 occurrences. By the looks of it, several common triples contribute to connecting most of the triples. This problem is a significant weakness in the algorithm, and we should have a general solution for this.

Rank	Subject	Predicate	Object	Score	Touched
0	CID129847363	has_parent	CID28780	74.878	334
1	CID67720914	has_parent	CID4101	74.878	334
2	CID20153713	has_parent	CID6579	74.878	334
		...			
140813	T690973	type	Term	74.878	334
140814	CID66608653	has_parent	CID10342051	10.592	2
		...			

Table 6.7: Results from the Basic Scoring algorithm without the object SmallMolecule.

6.3 Solution to Trivial Entities Connecting all Triples

As we can see in Section 6.2.1, the high connectivity in the graph is breaking the method, and we need as discussed a general solution to the problem. The following solutions do not handle all connections similar. For instance, a triple connected to a toxic chemical by many steps can be less likely to to be discriminating, compared to close triples.

For the following approaches, we applied the basic scoring algorithm described above, but switched out the `crawl` function with the new crawl approach.

6.3.1 Remove Common Triples

One naive solution would be to remove triples that occur too often in the graph. *e.g.*, remove all connections that are shared by more than 10% of the graph. Alternatively, we could remove entities that connect too many entities from both poles (harmful and non-harmful chemicals).

With this solution, it can be hard to find the optimal boundary. Critical triples that are common among one of the poles could be ruled out. It remains a possibility that the graph would still be connected through multiple steps.

This is not an adequate general solution. A solution like this would be quite graph dependent and not work in every case. The method is thus not explored further.

6.3.2 Limited Step Crawl (LSC)

In this section, we introduce crawls which are limited to a specific depth. The strategy of these approaches is to prioritize the local triples and ignore the higher hierarchy.

One Step Crawl

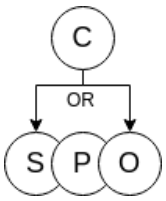


Figure 6.6: Crawl with one step. C is a chemical in the training data, and s,p,o is the triple receiving the score (*i.e.*, the LC50 concentration value minus the median) from the chemical.

The One Step Crawl is a somewhat restricted method. Only triples containing the chemical as object or subject are given the score from the chemical. We can see in Figure 6.6 how the scoring is limited to only direct connected triples in the graph.

Rank	Subject	Predicate	Object	Score	Touched
0	CID66608653	has_parent	CID10342051	5.296	1
1	CID70980984	has_parent	CID10342051	5.296	1
2	CID70288859	has_parent	CID10342051	5.296	1
3	CID88106247	has_parent	CID10342051	5.296	1
4	CID10342051	type	CHEBI_39346	5.296	1
5	CID11497855	has_parent	CID10342051	5.296	1
6	CID70013000	has_parent	CID10342051	5.296	1
7	CID10342051	type	C65672	5.296	1
8	CID3224	type	CHEBI_4791	4.519	1
9	CID88477627	has_parent	CID3224	4.519	1
10	CID22439579	has_parent	CID3224	4.519	1
...					

Table 6.8: Results from the Limited Steps Crawl algorithm .

However, only 23760, 16% of the triples in the graph, received a score above 0. We observed as well that all the scored triples are only affected by one chemical each. Hence, none of the chemicals in the training data is connected directly in the graph. This crawl is too limiting on this dataset.

$$\{s, p, o | (s, p, o) \in \mathbb{K}) \wedge s, o \in \mathbb{T} \wedge s \neq o\} = \emptyset$$

Figure 6.7: The symbol $\mathbb{K}$ represent the KG, and $\mathbb{T}$ the training set. The notation describes that there exist no triple in the KG, where both the subject and the object are from the training data. Hence, no chemicals we train on are directly connected in the KG

Two Step Crawl

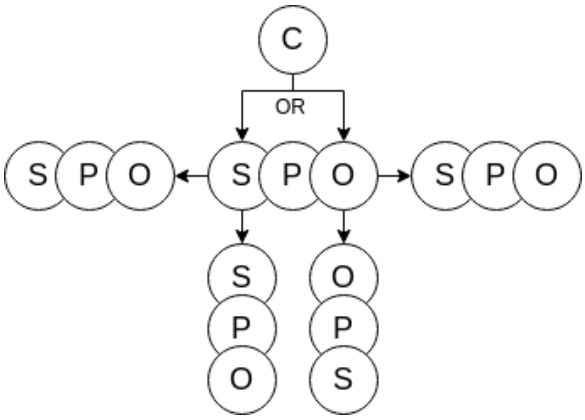


Figure 6.8: Crawl with two steps.

The One Step Crawl was in our case too restrictive to provide any useful information. The next strategy, however, takes the crawl one step deeper. The triples containing the chemical are further searched for connected triples, as we see in Figure 6.8. This extra step includes the shared attributes among the chemicals.

In Section 6.2 we mentioned an example of chemicals with the attribute insecticides. We can see in Figure 6.9 how both triples are reached, although only one of the triples contains a chemical from the training data.

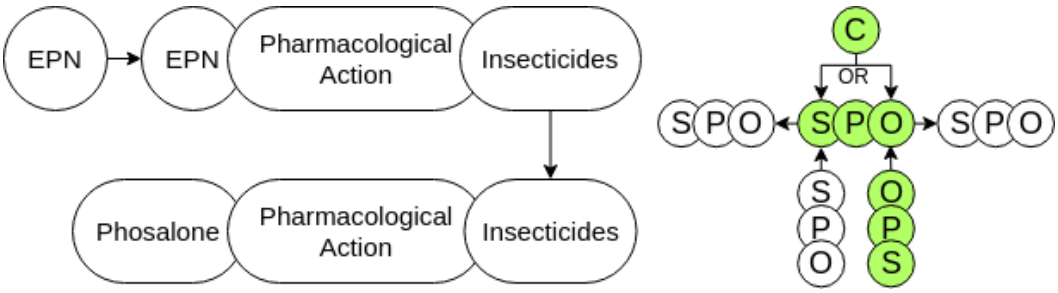


Figure 6.9: EPN: short for O-Ethyl O-(4-nitrophenyl) phenylphosphonothioate.

Table 6.9 presents the most discriminate triples according to this method. The identifier for insecticides, D007306, are present either as an object or subject in all of these triples. Investigating further, we see that all of the 85 triples containing relation to insecticides are ranked at the top.

The observation was a bit surprising as the chemicals categorized as insecticides were found to be less toxic than average. How insecticides affect the different organisms, it is not designed to harm, is outside the scope of this project. Although, it is interesting that the two-step method gave these triples such a high ranking. However, 54 chemicals out of all the 766 chemicals ($\approx$7%) we use in this project are classified as insecticides, which is a relatively large segment.

Rank	Subject	Predicate	Object	Score	Touched
0	D007306	seeAlso	D002800	81.379	61
1	D002800	seeAlso	D007306	81.379	61
2	D007306	AQ	Q000008	71.932	149
			...		
19	D007306	AQ	Q000528	71.932	149
20	D007306	AQ	Q000097	70.405	146
			...		
24	D007306	AQ	Q000276	70.405	146
25	D007306	AQ	Q000037	69.885	147
26	D007306	BD	D010575	68.455	45
27	CID3224	PA	D007306	67.638	43
28	D011722	PA	D007306	67.291	43
29	CID3347	PA	D007306	67.291	43
30	CID6758	PA	D007306	66.972	43
31	CID4115	PA	D007306	66.761	43
			...		

Table 6.9: Results from the Two Step Crawl algorithm. The predicates have the abbreviations: PA: Pharmacological Action, AQ: Allowable Qualifier and BD: Broader Descriptor.

About half of the graph, 62551 triples to be precise, are visited. However, we do observe that 6666 and 21430 triples have gotten only one or two touches, respectively. We could argue that there is a lower possibility that a chemical

is connected in the test data when having so few connections in the training data[1]. Hence, filtering these triples out could be an option if a strict scoring approach is preferred. We have nevertheless chosen not to neglect these, in concern of missing important information

6.3.3 Directed Crawl (DRC)

Simple Directed Crawl

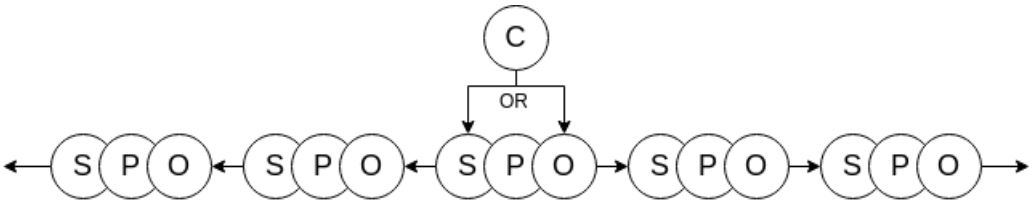


Figure 6.10: Simple Directed Crawl.

The previous methods take the local environment of the chemical into account. However, it does not utilize the hierarchy of the graph to the fullest, but this approach does. It will crawl to the top and the bottom, to include all super- and sub-classes. This method does not visit the entire graph because it only crawls in one direction, either from subject to object or object to subject, as we see in Figure 6.10.

This strategy will bypass the problem of having every triple connected, but as the limited step crawls, leave out information. The example we have earlier discussed with the insecticides will not be taken advantage of in this approach. However, in some graph structures, perhaps this one as well, the hierarchy could be more characteristic than the local environment. Therefore this approach is further explored.

[1]This is, of course, dependent on the structure of the KG

Rank	Subject	Predicate	Object	Score	Touched
0	D006838	AQ	Q000378	66.928	67
			...		
31	D006838	AQ	Q000600	66.928	67
32	D006571	AQ	Q000008	65.382	140
			...		
63	D03	type	TreeNumber	65.382	140
64	D02.455	type	TreeNumber	64.828	71
65	D02.455	PTN	D02	64.828	71
66	D004786	AQ	Q000032	63.479	130
			...		
173	D010575	AQ	Q000600	63.479	130
174	D005659	AQ	Q000600	63.139	129
			...		

Table 6.10: Results from the Simple Directed Crawl algorithm. The predicates have the abbreviations: AQ: Allowable Qualifier and PTN: Parent Tree Number.

Directed Crawl with back Steps

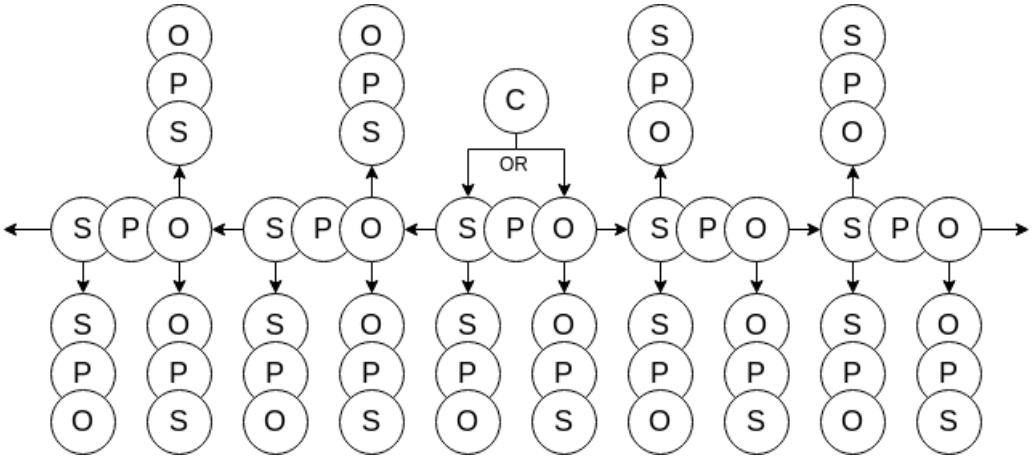


Figure 6.11: Directed with back steps.

The LSC and the Simple DRC focuses on two different aspects of the graph. Both the strategies consequently neglect the aspects the other strategy considers. A strategy that combines the benefits from both approaches could conceivably be the best solution. Hence, we introduce a mix of them both: a

DRC with one back-step. This way, local information along the hierarchy is included while the crawl remains somehow restricted.

Rank	Subject	Predicate	Object	Score	Touched
0	M0010687	preferredTerm	T020636	133.856	134
1	D006838	preferredTerm	T020636	133.856	134
2	D006838	preferredConcept	M0010687	133.856	134
3	D006838	treeNumber	D02.455	131.756	138
4	M0010290	preferredTerm	T019788	130.764	280
		...			
209	D035141	allowableQualifier	Q000331	115.280	252
		...			
3880	D002906	allowableQualifier	Q000469	115.280	252
		...			

Table 6.11: Results from the DRC with back Steps algorithm.

We recognize in Table 6.11 that from rank 209 to rank 3880, 3671 triples share the same amount of touch and have the same score. We suspect that this approach is too close to crawling the whole graph and that this strategy is inadequately to separate the triples properly.

Directed Crawl with back Step on first Step

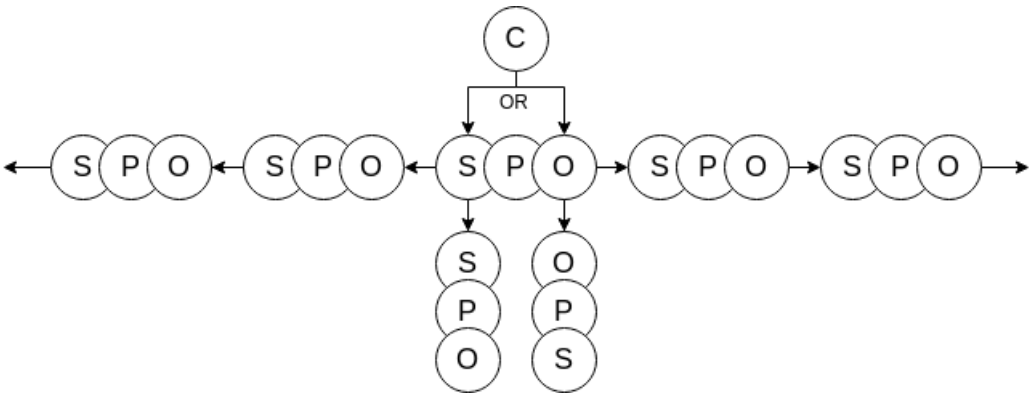


Figure 6.12: DRC with back step on first step.

This strategy is again a compromise, but now between the two previous methods. One could assume that the local triples close to the chemical are more

relevant to the ones of the super- and sub-classes. In this crawl, we only take one step back at the first step. We can define this as a union between the Two Step Crawl and the Simple Directed Crawl.

Rank	Subject	Predicate	Object	Score	Touched
190	M0000913	preferredTerm	T001785	57.990	126
		...			
79445	M0543318	preferredTerm	T765305	57.305	125
		...			
83739	D001304	allowableQualifier	Q000191	37.897	159
		...			
99122	D012893	allowableQualifier	Q000276	37.490	157
		...			
103040	D016769	allowableQualifier	Q000097	37.490	157
103041	D000438	allowableQualifier	Q000037	36.970	158
		...			
103304	D014166	allowableQualifier	Q000037	36.970	158
		...			

Table 6.12: Results from the Directed Crawl with Back step on first Step algorithm.

We can still see similar scores being shared by multiple triples. However, not to the extent as the previous method. Even though many of the scores are similar, we can see some variations between most of them. Besides, the clusters themselves, have a higher difference separating them.

6.3.4 Descending Influence Crawl (DIC)

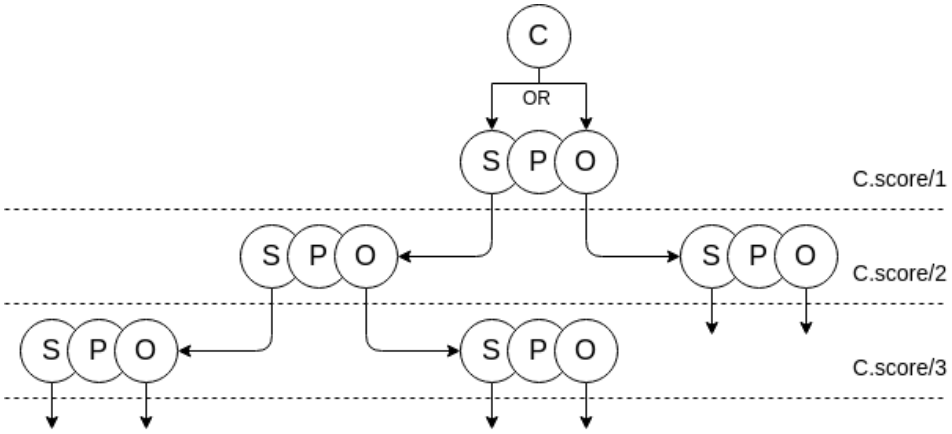


Figure 6.13: Crawl with Descending Influence. Only the directed triples are drawn to simplify the illustration. However, the subject and objects can be connected to other subjects and objects, respectively (Like we see in Figure 6.1).

The previous approaches we have introduced cuts the crawling short in order to limit the influence from each triple. Triples are either in relevance to the chemical or not, with no middle ground. However, we believe that the relevance of a triple descends the further away it is from the chemical.

With this assumption in mind, we introduce an approach we call Descending Influence. We divide the score by the number of steps from the chemical before supplying it to the triple. Every chemical will touch most triples. However, the influence will vary.

Rank	Subject	Predicate	Object	Score	Touched
0	CID3347	PA	D007306	38.0256	456
1	CID38779	PA	D007306	36.9928	456
2	CID6950	PA	D007306	36.8743	456
			...		
69163	D011859	AQ	Q000941	5.6471	456
69164	D000071199	AQ	Q000502	5.6465	456
69165	D033101	AQ	Q000706	5.6459	456
			...		
144308	CID5460998	has_parent	CID289	1.6910	456
144309	CID13529	type	CHEBI_63104	1.6905	2
			...		

Table 6.13: Results from the Descending Influence algorithm. The predicates have the abbreviations: PA: Pharmacological Action and AQ: Allowable Qualifier.

Chapter 7
The Execution

In this chapter, we will discuss some of our procedures and choices regarding the experiments. We deliberate on which triple scoring algorithms we decided to examine, how we passed the data to our models, and lastly how many times we repeated the various training processes.
We have chosen to experiment with the following triple scoring strategies.

- **Baseline Method (BLM)**
  The BLM utilises the whole KG and handles all the triples equally. The results from this are mainly used to compare other approaches.

- **Only CID-mapped (OCM)**
  We described in Chapter 5 how we collect the data from two sources, the ChEMBL and the MeSH datasets. All the triples in the ChEMBL dataset have either the triple or the subject mapped to a CID. We showed how this dataset alone did not provide enough information and complimented it with the MeSH data. However, this dataset has only a smaller subset mapped to CID, and the most mapped to other identifiers. We question if the new CID-mapped triples alone provide the knowledge graph with the valuable information. Therefore we include this approach in our study.

- **Limited Step Crawl (LSC)**
  When limiting the crawl with numbers of step, the Two Step Crawl resembled a great place to draw the line. The One Step Crawl did not connect any of the chemicals while introducing more steps could connect the graph too strongly again.

- **Directed Crawl (DRC)**
  We introduced three different strategies for the Directed Drawl. They are similar in their approach but includes local triples in different ways. The Simple Directed Crawl neglects these complete, while the Directed Crawl with Back Steps tends to include too many triples and reminds us of the original baseline method, where all the triples are handled equally. The last strategy with back step only on first step seems to be the best hybrid to combine the local and hierarchy information. Hence, we have chosen only to examine this method.

- **Descending Influence Crawl (DIC)**
  The last strategy we have chosen to include is Descending Influence. It employs all triples but handles them differently based on the distance from the chemical. This will, to some extent, prioritize the local context, but not dismiss the higher parts of the hierarchy completely.

7.1 Preparing the Data

7.1.1 Generating the Triple Output

Triples we input to the model needs to have suitable output to be used during training. The embeddings are in our model implemented with a Sigmoid activation layer for the output layer. The Sigmoid ensures that the result is in the range between zero and one. The number can be interpreted as the probability of triple existing in KG. Strictly, in this work, we tie this to the importance of the triple.

We mentioned in Section 2.4.1 how the embeddings need to be supplied with negative triples to prevent the trivial solution. We generated negative triples by randomly combining entities[1] as subjects and objects. If the triple does not already exist in the KG, it will be added with a score of zero. When it comes to the existing triples, we use three distinct approaches to give them a suitable output.

[1]Entities is referring to the union of subject and objects.

- **Binary (Bin.)**

  For the BLM and the OCM approaches, we take a binary approach. Since we only know that the triples exist and nothing more, they are given the score 1.

- **Normalization (Nor.)**

  In the second approach, we normalize the scores between 0.5 and 1. The reason we sat the lowest value to 0.5, is because the triples who are irrelevant does still exist, and differ from false triples. This method is used for the LSC, DRC and the DIC strategies.

- **Average (Avg.)**

  The last method is similar to the normalized method. The only difference is that the score is divided on the number of touches before the normalization. Calculating the average before the normalization will avoid the problem of triples connected to many less-toxic chemicals could exceed triples connected to a few highly-toxic chemicals. We have tested this on the same approaches as we did with the second approach.

  One exception is for the DIC. Since the received scores are quotients of their own, dividing on the number of touches will not be correct. The scores received from a chemical far away will then have a negative impact, despite it having a high concentration. Hence, we apply the Weighted Average (WA) instead.

7.1.2 Balancing the Input

Our KG-model takes in two datasets when training, the KG and the training data. They must be of equal length for both of them to be present through every forward and backpropagation. The training data set is smaller than the KG. We replicate arbitrary instances in the training data to compensate for the gap.

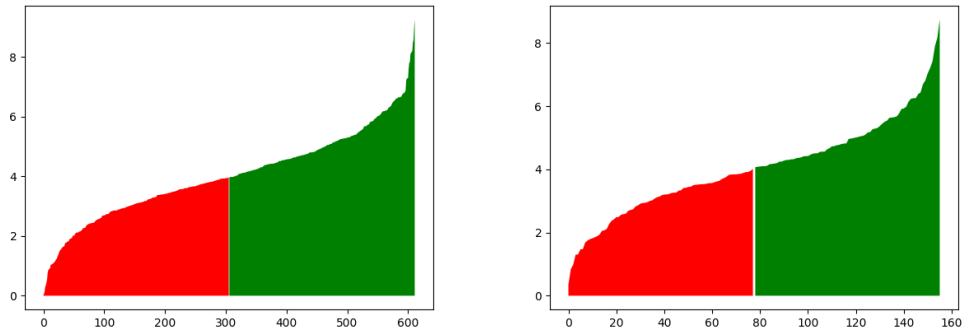
Hence, when using shorter graphs, the epochs will include fewer instances from the training data, compared to more extended graphs. This can have

an influence on the result in terms of overfitting and underfitting. Although we have solutions for overfitting, comparing the KG-model to the ON-model, will evaluate how they perform given the same circumstances.

7.1.3 Converting the Concentrations to Binary

The ecotoxicology effects are measured in concentration (mg/L). However, to simplify the problem to a classification problem, we divide chemicals into two classes, toxic and non-toxic.

The data are split at the median, resulting in equal parts of toxic and non-toxic chemicals. In both the training and the test data, the median is at about 3.9 mg/L. It is conceivable that it exists a more optimal division than such; however, in the scope of this project, we will use this simplification.



(a) The division on the training data (b) The division on the test data

Figure 7.1: Shows where the data separates at the median in linear-scale.

7.2 Repetitions

7.2.1 Number of Epochs

The term epoch means training iteration. When the whole training data is passed through the model and backpropagated, we call it an epoch. This can, and usually should be repeated multiple times. If the number of epochs is too low, we will get an underfitted model, whereas if the number of epochs is too high, the model can be overfitted. It is challenging to find the perfect number of epochs needed to get the best generalization from the model. The balance is not on a fixed number and can vary for each run. However, as we

have discussed in Section 2.3.2, methods to avoid overfitting exists. We have therefore decided to go for a relatively high number of epochs and settled on 100. To work against overfitting, we used the following strategies.

7.2.2 Overfitting

Our model uses, as illustrated in Figure 4.2, dropout on each hidden layer. This is a method to avoid overfitting, where arbitrary nodes in the network are left out during the training.

As we further deliberate in Chapter 9, the result from just using dropout were less satisfying than what we expected. Although dropout assists to avoid overfitting, we still suspect overfitting to be a potential cause for the low values. Hence, we ran the models with early stopping in addition.

7.2.3 Number of Runs

ML is a stochastic method, where the results can vary based on initial conditions[2]. Consequently, we need to perform multiple executions to get a representative average — the more runs, the more reliable result. However, the process of training the network is, as discussed, time-consuming.

We have chosen two embedding algorithms, two strategies against overfitting, and five scoring approaches, where three of them have two ways to incorporate the scored triples. That gives us a total of 32 combinations.

Although increasing the number of runs per combination give us a more accurate average, it also increases the number of runs rapidly. We have, therefore, decided to do seven runs per combination, which leaves us with a total of 224 runs.

[2]We use random initialization of model weights.

Chapter 8
Results

In this chapter, we present the results from our experiments with explanations on how to read and interpret them. We observe that the Limited Step Crawl (LSC) with average and the Directed Crawl (DRC) normalized, both using DistMult, perform relatively good in all metrics. The Only CID-mapped (OCM) with TransE have also shown an improvement in all the metrics, except in one (p-value). However, the baseline, along with the other approaches, does not perform a lot better than the random classifier.

8.1 Evaluation Metrics

This section describes the evaluation metrics we use to compare the different approaches, to provide the reader with sufficient background knowledge to be capable of interpreting the results.

8.1.1 Precision and Recall

We can divide all the results into four categories.

True Positive (TP)	False Positive (FP)
False Negative (FN)	True Negatives (TN)

The true positives and the true negatives on the leading diagonal are where the prediction is correct and should be greater than the two others. From this, we can calculate the accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

The accuracy can be explained as how many correct predictions we received in correlation to the false. However, the metric does not describe the results further. It does not separate between the positive and the negative results. We can, nevertheless, use a complementary pair called precision and recall. The ratio precision is how many of the positive samples were, in fact, relevant. Recall, on the other side, is measuring how many of the relevant positive samples we caught, as opposed to not. The two metrics are inversely connected and can be united into one metric [51, p. 23].

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

F1 is considering all of the four outcomes by utilising precision and recall. It does, however, conceal details in our area of concern. In ecotoxicology effect prediction, it is essential to capture as many of the true positives as possible, despite the increase in false positives. Hence, we want as few false negatives as conceivable to avoid undetected hazards. The recall can measure this, as it reflects how many of the relevant elements we manage to discover.

8.1.2 ROC and AUC

ROC is an abbreviation for Receiver Operator Characteristic and is displayed as a curve where true positives are on one axis, and false positives on the other, as we vary the decision threshold.

The ROC curve must lie above the diagonal from the line from (0,0) to (1,1), for it to be better than random, at that specific point. The higher the curve, the better the prediction. To compare different classifiers, AUC, short for Area Under Curve, can be calculated. The bigger the area, the better the classification [51, p. 24].

It is customarily to use AUC when comparing classifiers. However, when it comes to ecotoxicology effect prediction, we value true positives, despite the increase in false positives. Hence, it is reasonable for us to focus on the upper right of the curves, which is lined out in Figure 8.1.

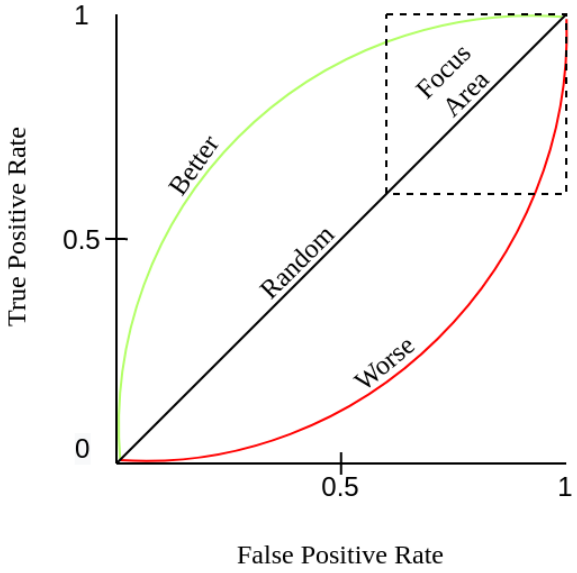


Figure 8.1: ROC curve.

8.1.3 Probability Value

Probability value (p-value) is a statistical model commonly used as a test of significance for the findings in research. Ronald Fisher, how is considered as the pioneer, interpreted the value as an index of evidence against the null hypothesis (*i.e.*, no difference in the datasets) [50].

It has been common to interpret the p-value as the probability of the findings to be the result of chance. As a consequence, it is believed that the remaining rate is the probability of the groups to differ. This is, however, not entirely correct [22]. A high p-value is not an indication for the groups to be similar. Instead, it should be interpreted as a lack of evidence for the groups to differentiate, in this specific observation [56]. It does not say anything about the probability of groups to be equal [83].

The American Statistical Association (ASA) have released statements about misconceptions and misuse of the p-value. Among others, they mention how the p-value is not a measurement of the effect or importance of a result. It does not say how much the groups separate, but can rather present evidence if they do so [83]. They also emphasise that the p-value alone should not be the ground for the conclusion of the hypothesis [83].

> "The p-value was never intended to be a substitute for scientific reasoning" - Ron Wasserstein, the ASA's executive director [83]

Ronald Fisher introduced <5% as a standard threshold to conclude that the findings were of statistical significance. He later clarifies that this is not an absolute rule, but rather a suggested cutoff. The cutoff has been perceived as standard, especially in medical research, but if the p-value exceeds 5%, we can still argue that there are tendencies toward statistical significance [29].

We collected, from all our seven runs, the F1 values from the KG- and the ON-model in separate sets. The two sets were then used in the calculation to find the p-value. The intent is to provide evidence that the sets are different and conceivably strengthen the hypothesis that the KG-model outperformed the ON-model or vice versa.

8.2 Without Early Stopping

Before introducing Table 8.1 with the results, we will describe how the columns in the table are read. In the first column, "Approach", we find the different approaches to cut or prioritize the graph, that we decided to include, at the start of Chapter 7. The second column, labelled "Scoring", refers to the distinct strategies to use the score of the triple in the model. We deliberated on this in Section 7.1.1. The third column, "Embedding", specifies which embedding method that was used. As we decided in Section 4.3.1, we used two embedding methods, DistMult and TransE. They are abbreviated in the table as DM and TE, respectively.

The following columns: "F1", "p-value" and "AUC", are referring to the metrics (described in Section 8.1) from the KG-models. "Δ F1" and "Δ AUC" are the difference between the ON- and the KG-model. If the figure is positive, it means that the KG model outperformed the ON-model and visa versa.

The table is colour coded with yellow and green. In columns "Δ F1" and "Δ AUC", we marked the results that outperformed the baseline method, respectively to the embedding method applied, as green. The p-values marked yellow, are relatively close to statistically significant.

Approach	Scoring	Emdedding	F1 (%)	Δ F1 (%)	p-value (%)	AUC (%)	Δ AUC (%)	Recall (%)	Δ Recall (%)
BLM	Bin.	DM	49.7	+3.4	40.9	52.4	+2.5	43.5	−6.8
BLM	Bin.	TE	47.6	+1.2	79.4	52.2	+2.7	46.0	+1.7
OCM	Bin.	DM	41.6	−10	21.2	51.0	−4.4	39.8	−11.6
OCM	Bin.	TE	56.9	+6.4	44.8	58.5	+9.1	67.3	+15.3
LSC	Nor.	DM	47.4	−4.6	23.7	49.3	−1.3	47.2	−7.0
LSC	Nor.	TE	52.8	+2.1	46.9	51.6	+0.1	54.7	+3.5
LSC	Avg.	DM	56.4	+5.3	16.9	54.2	+6.1	60.2	+6.0
LSC	Avg.	TE	49.0	−0.7	86.4	49.8	+0.6	48.9	−1.9
DRC	Nor.	DM	51.9	+3,3	44.0	51.7	+1.4	53.6	+4.6
DRC	Nor.	TE	52.1	+0.5	87.9	52.2	+0.3	53.4	+1.0
DRC	Avg.	DM	46.8	−0.3	96.1	49.7	−1.7	44.7	−1.0
DRC	Avg.	TE	51.3	+1.5	66.9	51.8	+3.3	53.0	+2.1
DIC	Nor.	DM	55.9	+7.1	6.7	55.3	+3.8	58.8	+9.5
DIC	Nor.	TE	48.2	−1.4	65.6	50.4	+1.0	46.6	−3.5
DIC	WA	DM	49.4	−1.2	78.2	51.9	+2.3	50.5	−2.9
DIC	WA	TE	53.0	+2.4	42.8	51.1	+1.3	55.3	+1.9

Table 8.1: Results without early stopping.

8.2.1 ROC Comparison

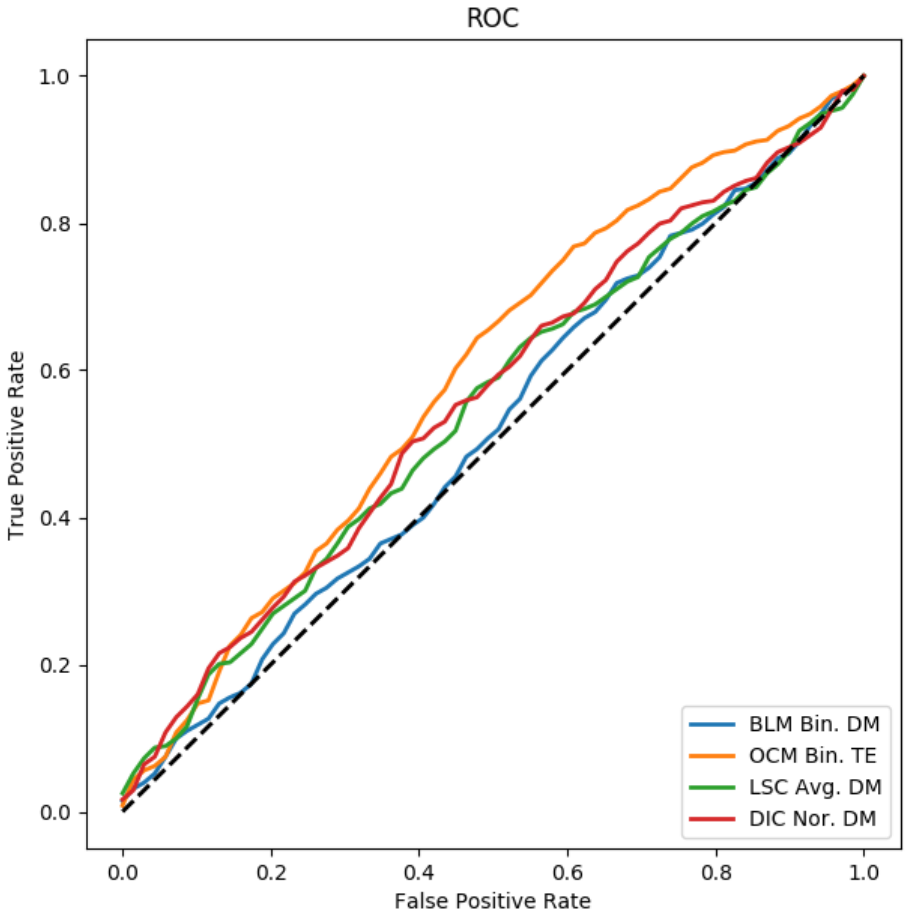


Figure 8.2: ROC comparison using macro average.

Figure 8.2 is a plot of the ROC curves for the three best performing classifiers, along with Baseline Method. For the BLM, we used DistMult. This has both the best performance compared to using TransE and are used by two out of the three best classifiers.

At the start of the curves, we see that BLM is lying close to the diagonal random classifier. The other three are all above at a similar height. At the middle part of the graph, we can the pattern continue, except for the Only CID-mapped that raises even more. Towards the end of the curve, we observe the Descending Influence Crawl and Limited Step Crawl fall to the level of

the BLM. This is easier to see in Figure 8.3, where we zoomed in on the top of the graph.

All the three classifiers are better, or equal, classifiers than the baseline, in most areas in the figure. However, we see that the Only CID-mapped is giving the best results.

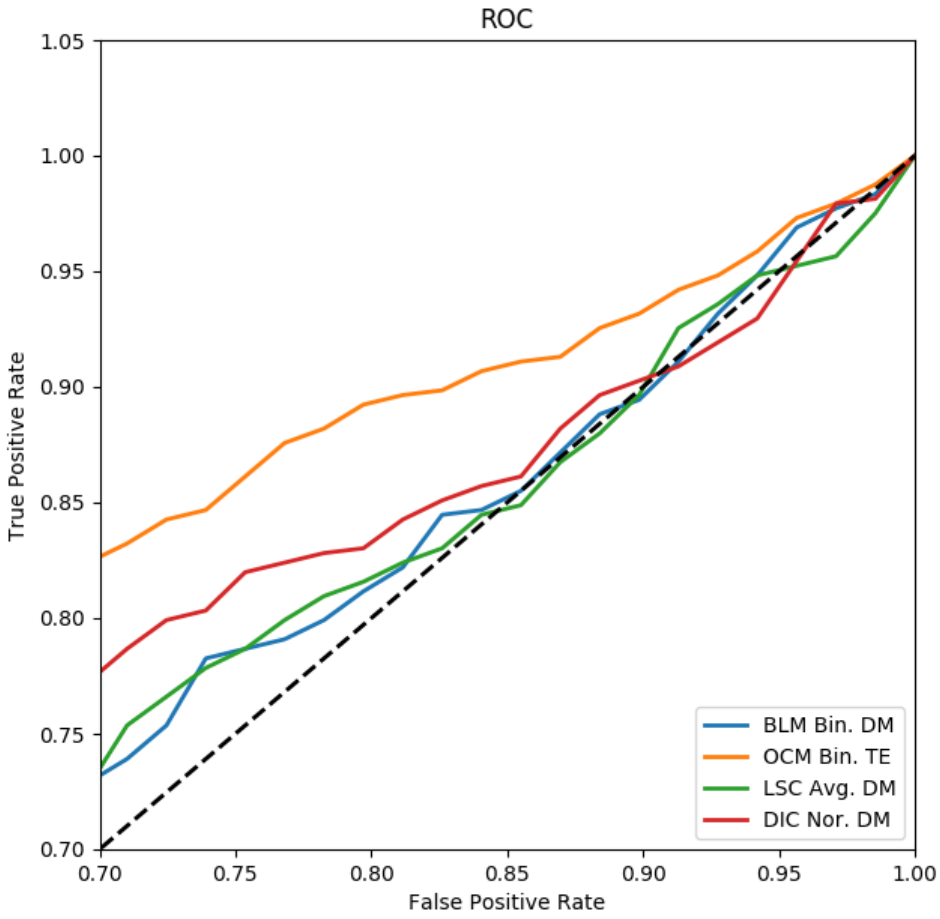


Figure 8.3: ROC comparison using macro average, zoomed to our area of focus.

8.3 With Early Stopping

In Table 8.2, we present the results when using early stopping in addition to dropout. The columns and colour coding are equal to Table 8.1.

Approach	Scoring	Emdedding	F1 (%)	Δ F1 (%)	p-value (%)	AUC (%)	Δ AUC (%)	Recall (%)	Δ Recall (%)
BLM	Bin.	DM	51.1	+13.3	21.6	49.3	+0.0	58.4	+17.2
BLM	Bin.	TE	45.9	+5.0	57.9	47.9	−1.9	45.5	+1.2
OCM	Bin.	DM	41.0	−2.6	80.6	50.7	−2.6	41.2.0	−2.5
OCM	Bin.	TE	49.7	+15.3	21.1	48.6	+0.5	61.5	+26.3
LSC	Nor.	DM	41.7	+5.7	63.5	51.3	−1.4	45.3	+6.4
LSC	Nor.	TE	47.2	+10.6	35.0	49.1	+1.4	56.9	+19.7
LSC	Avg.	DM	52.6	+0.9	91.2	50.3	−0.4	65.2	+7.2
LSC	Avg.	TE	41.4	+10.0	48.5	52.5	+3.9	48.9	+15.5
DRC	Nor.	DM	42.8	+11.7	29.7	47.8	+0.3	48.9	+24.6
DRC	Nor.	TE	46.9	+1.7	81.8	49.6	+1.6	49.9	+1.9
DRC	Avg.	DM	45.8	+8.8	43.5	50.0	+0.5	48.0	+6.2
DRC	Avg.	TE	41.4	−3.3	76.9	48.4	−1.5	48.2	+2.1
DIC	Nor.	DM	48.0	−3.6	71.6	51.5	+2.5	57.8	−2.5
DIC	Nor.	TE	47.7	+5.5	58.9	49.2	+0.9	55.9	+13.5
DIC	WA	DM	52.3	+3.7	73.5	46.6	−4.0	65.2	+1.2
DIC	WA	TE	49.8	+13.8	20.9	47.4	−1.3	55.1	+16.6

Table 8.2: Results with early stopping.

Chapter 9
Discussion

In this chapter, we will discuss the results from our experiments, and deliberate on feasible explanations for the patterns we observe. We consider the characteristics of the embedding methods, what data the various crawling approaches prioritize, and more.

9.1 Challenges with the Crawling Methods

We face multiple challenges with these approach. The following paragraphs will mention a few of the most notable obstacles we encounter.

Problem: If some of the test chemicals do not have any connected triples with the train data, the method does not consider relevant triples for those chemicals in particular.
Answer: These triples would not have any impact since they are not relevant to the training and is irrelevant to the training even if it contains essential information.

Problem: Even though triples that occur in many iterations from the crawl can be a sign of relevance, it can also overshadow other triples that do not appear that often if they get an unrealistic high score.
Solution: A alternative is to use the average on the scored triples instead of only normalizing the score. Both methods are assessed.

Problem: Triples that are non-discriminating can get a high score using our methods if not equally represented in both poles.

Answer: This is an issue that can falsely find discriminating triples, and can be problematic when applying this algorithm to specifically rank triples. However, when using it in the ML model, it can still be beneficial. The BLM gives all the triples a score of 1. This method is filtering out noise from the KG. If many triples are falsely given a high score, the effect of this method could decline. However, the bigger problem is when essential triples do not get the score they justify, as this could lead to a loss of discriminating information.

9.2 The Unsatisfying P-Values

The first thing we notice in the results were the unsatisfying p-values. They are all in the range of 6.7% and 96.1%, where most are above 20%. We can also see that the value for the BLM is just as unsatisfying, with 40.9% and 79.4% when using DistMult and TransE, respectively. However, this does not mean that we should reject the results. What it implies is that the results were not proven to be statistically significant, according to the p-value.

As we mentioned in Section 8.1.3, we should not rely on the p-value alone and substitute it from scientific reasoning. We also expect variation within the results as this is a non-deterministic method and besides a difficult learning problem. We do, however, see two rows in Table 8.1 that stands out (marked in yellow). Limited Step Crawl has a p-value of 16.9% when used with average and DistMult. In another context, this could have been perceived as an insignificant finding, but the value is relatively decent compared to the other values. As we mentioned in Section 8.1.3, The 5% cutoff is just a suggestion. The other row is the Descending Influence Crawl also used with DistMult, only normalized. Its p-value is 6.7% and can be interpreted as close to a statistical significance.

9.3 The Low Values in General

None of the values was particularly exceptional, compared to other prediction cases. The AUC is a metric that can be used to compare which classifier is the best. A classifier predicting by random choice would get a straight diagonal line through the ROC, and therefore, get an AUC value of 50%. All of the results without early stopping have AUC values between 49.3% and 58.5%.

As we can see in Figure 8.1, the BLM has an AUC of only 52.4% and 52.2% using DistMult and TransE, respectively.

Although the results are a bit under our expectation, we must recognise that this is a challenging learning problem and be gratified with only small improvements. The results were not statistically significant, but the problem may not be in our crawling approach. The bad performance for the BLM does also indicate this. The problem can be in the KG embedding approach or the KG itself. However, we can still see interesting patterns in the results.

9.4 The Results without Early Stopping

We can see three approaches that outperformed the baseline classifier in all the different metrics.

- Only CID-mapped (OCM) - Binary - TransE
- Limited Step Crawl (LSC) - Average - DistMult
- Descending Influence Crawl (DIC) - Normalized - DistMult

As we discussed in the previous section (Section 9.2), the two last-mentioned methods, were also the ones with the best p-values.

The Directed Crawl (DRC) with average and TransE are close to being in this category as well, had it not been for a slighter lower AUC. We do consider the Δ AUC over the AUC, as it is a comparison under the same circumstances. However, the other metrics do not differentiate much. Neither do the BLM results nor do the KG-model to the ON-model. Hence, we disregard this as an outperforming approach.

Local Area over the Higher Hierarchy

A pattern we see in the results is that the approaches prioritizing local triples perform better than the DRC looking at all sub and superclasses. As we mentioned in Section 5, the MeSH dataset does contain hierarchy information on the chemicals. This can indicate two things. Either the chemicals used in the experiments unexpectedly lacks discriminating hierarchical information, or this has little to say when it comes to the toxicity of the chemicals.

However, the DRC we used does not look back other than on the first step. As we showed in Section 6.3.3, that would lead to a too connected graph, similar to using all the connections. If all the chemicals are on the same level in the hierarchy, we will reach siblings, parents and grandparents but not cousins. This could give the classes higher in the hierarchy a bias compared to the neighbouring. The embeddings, especially TransE (see Figure 2.17) should handle such relations. Although, if none of the cousin's siblings is included in other training sets, the connection from the sibling to the common grandparents can get weak. We are not aware of how big of an influence this has. Most of the approaches include all the triples, but give the poorly scored an output toward, but not lower than, 0.5.

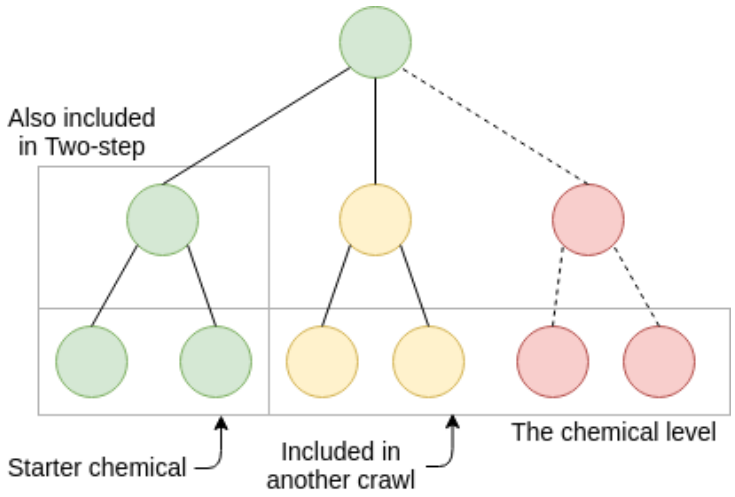


Figure 9.1: Illustrates what the DRC includes compared to the LSC, under the assumption that all chemicals are on the same level in the hierarchy. The nodes marked green are the once visited from this crawl, the yellow nodes are the one visited form another crawl, and the red is never visited. The dotted lines indicate a weak link.

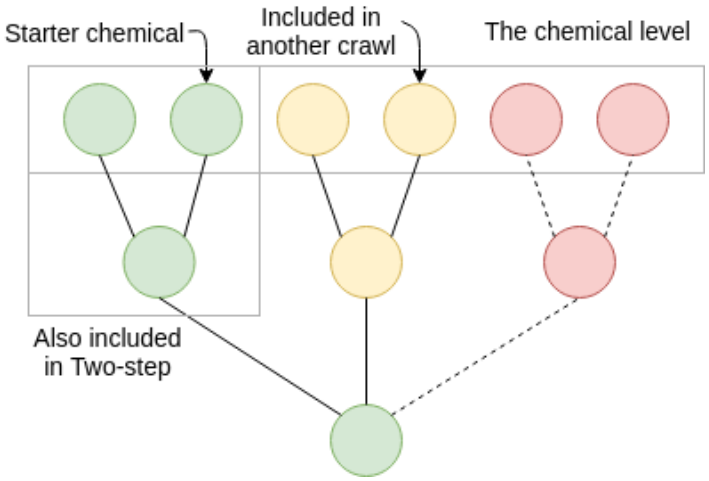


Figure 9.2: The same principle applies for subclasses of the chemicals on the same level.

The DRC is a union between the LSC and the Simple DRC (*i.e.*, one step back on the first step). Nevertheless, it performs worse than the LSC. This could mean that the super-super-classes (*i.e.*, grandparents), and the sub-sub-classes (*i.e.*, grandchildren), hold little or no discriminating information.

DistMult over TransE

The best approaches, except the OCM, performs best with DistMult compared to TransE. We discovered in Section 3.9, how TransE performs poorly in complex relation categories (*e.g.*, one-many, many-one and may-many). Hence the reason might be that most local relations in our data are of this complexity and not as simple as Figure 2.17 in Section 2.4.1 depicts. It could resemble, judging by our results, that DistMult is better at handling such relations. As we also discussed in Section 3.9, DistMult is performing twice as good as TransE with these complex relation categories, which lines up good with our results.

We do, however, observe that TransE is performing better than DistMult in some approaches (e.g. DRC with average and OCM binary). The DRC has not as significant results as the other approaches, but we still see a small improvement of TransE from DistMult. We could assume that nodes higher and lower in the tree, has less complex relations, especially on one of the sides (*i.e.*, none or few of many-many relations), and that TransE handles these better.

As for the only OCM, we believe that a lot of the complicated relations are due to the triples in the MeSH dataset from other vocabularies, not directly connected to CID chemicals but mappings between other concepts. Few of the rows in MeSH have CID-mapped relations, but they alone could have been enough to supply the ChEMBL dataset. By removing the other triples, we avoid exposing TransE to noise (at least noise for TransE as it does not handle complex relations). However, the results of the OCM have a high p-value compared to the two other outperforming approaches. Hence, we should interpret the results with some scepticism.

LSC is Best with Average, while DIC only Normalized

For the two other approaches, we recognize that the DIC worked better with the scores only normalized, while the LSC, worked best by applying average on the scores first. We believe that by applying weighted average on the scores from the DIC, the effect of reducing the impact from certain chemicals becomes too great.

9.5 Results with Early Stopping

We observe more variation in the results using early stopping. Most of the runs, using this technique had only between eight and sixteen epochs. These are relatively low numbers. The research introducing DistMult [87] used 100 epochs for FB15k and 300 for WordNet (WN). Hence, we believe that the networks where underfitted, and that the results occur from random chance in the model weight initialization. The AUC values from the BLM indicate that it performs worse than random. We see high delta values in Table 8.2, however, they can be explained by the poor performance from ON-models. The KG-models results are still worse than random.

Early stopping can be challenging to apply in this problem, where we do not expect the performance to increase a lot between runs. What we could have done is to tweak the patience or the patience delta variable to include more epochs. The early stopping was included under the concern that dropout was not good enough to prevent overfitting. However, we conclude that early stopping is not suitable for this problem and that dropout is better used alone.

9.6 Explainability

The secondary objective of this master thesis project was to use the KGs for explainability. This approach does not provide a prediction specific explanation. However, the ranking of the triples, from the successful methods, can be of interest when determining its level of discrimination.

We mentioned in Section 6.3.2, how the LSC produced a ranking where the top 85 triples were connected to the identifier for insecticides. Surprisingly, we discovered that the chemicals classified as insecticides were less toxic than the average of the chemicals. To find the reason behind this is left out of the scope of this project. However, it is reasonable to assume that the label insecticide is relevant to the toxicity of the chemicals.

It is worth to mention that triples can receive good scores, despite them being non-discriminating. The results from the crawls should be interpreted as a filter, to remove uninteresting information.

9.7 Ethics

Collecting data in the field of ecotoxicology involves, in most cases, animal testing. As we rely on such data, it introduces ethical concerns, which are necessary to recognise. The data we use in this master thesis project is the concentration required to kill 50% of the population of fathead minnow. It is worth mentioning that the data used in this theses is public information, conducted for other purposes. The more the results from the experiments are used, the more it justifies the applied hazards for the test subjects.

We would also argue that the outcome of ecotoxicological research, in general, is balancing out the harm it causes. The objective is to preserve the wildlife by classifying toxic chemicals and avoid harmful substances from infiltrating natural habitats [26]. Although the experiments can be justified, the exposure should be kept to a minimum. Ecotoxicological effect prediction, the challenge explored in this master thesis, can be used in the ecological risk assessment pipeline to facilitate fewer and more explicit laboratory experiments [55].

Based on the arguments made, we would argue that the research is within a good ethical ground. The research of ecotoxicological effect prediction will both limit animal testing in the laboratories and assist in finding hazard chem-

icals to prevent harm to nature.

Part III
Conclusion and Future Work

Chapter 10
Conclusion

In this master thesis project, we have embodied weighted Knowledge Graphs with Machine Learning models using vector embeddings for ecotoxicology effect prediction. The weights of the KGs are constructed based on the triples connection to chemicals in the training data. To enable that, we introduce crawling algorithms that start from each chemical in the training data, and in their distinct ways, crawl the graph, scoring the triples based on the chemical toxicity. As the graph is connected, we propose various methods of limiting the crawling, to avoid the trivial solution where all the chemicals influence all the triples. In addition, we have performed a comprehensive evaluation and given compelling insight into the performance of KG embeddings in general.

We show that three of our approaches outperforms the baseline in all the metrics we use for evaluation. Even though the results are less significant than expected, we observe improvements to the prediction. The low metrics can be due to other factors like the KG embedding approach or the KG itself and does not necessarily describe our approach, especially since the baseline performed close to random.

When it comes to explainability, our approach brings us a small step closer to explainable models. The crawling and scoring of the graph can highlight discriminative triples by assigning then a high ranking.

We have justified the ethics behind this research and concluded that we are within a good ethical ground. While we only use publicly available data, the purpose of this project is to discover unobserved chemical hazards and evidently avoid toxic runoffs into nature.

In the future, we hope that this project can contribute to precise, comprehensive and intelligible predictions, in order to limit hazardous experiments

performed on test organisms.

10.1 Future Work

Explainability

We believe that a lot can be further investigated when it comes to the explainability, and perhaps a similar crawl strategy can be used to explain the individual predictions. By crawling from the predicted chemicals and collecting triples with a good score, suggestions can be provided. If we were to store the scored triples with positive and negative signs to represent non-toxic and toxic chemicals, respectively, we could focus on only relevant triples for the outcome. The prediction process would not conduct the suggestions, and the two crawls would be on each side of the system architecture. However, reasonable explanations could be presented.

Other Scoring Approaches

In this master thesis project, we have scored the triples based on the concentration of the connected chemicals. However, this is only one of many possible ways of using the training data to arrange the graph. Another approach is to use the training data after it is binary converted. Then all records have either the value 1 for toxic and 0 for non-toxic. If we for each triple had two counters, we could then crawling the graph from the chemicals, and keep track of how many toxic and non-toxic the triples connects to. This way, we can make decisions on how discriminating the triples is, based on its representation in the two poles. This is just an example to emphasis that there are many creative ways this can be achieved.

Other Species

Our experiments are confined to only concern the fathead minnow. However, our methods can easily be extended to include multiple species. It would be interesting to see how this can affect the results from the predictions and to compare the different ratings of the triples.

As we only consider one species, we have not included additional species information to our KG. However, such data are available in triples from TERA

and can conveniently be included. Thus, chemical effects can be predicted based on the chemicals interactions with other similar species. This could potentially improve the prediction. However, the increased size of the KG would result in more expensive epochs and require more computational power.

Other Applications

It would be interesting to see how the strategies introduced in this thesis would perform on other applications in other domains. Especially since traditional KG training sets (*e.g.*, FreeBase), do not include corresponding training data, which strategies in this thesis utilise. The structure of the data is a significant factor to which of the various approaches proposed are the most beneficial. With less complex relations and a more straightforward hierarchy, both TransE and the Directed approach could have performed better.

Bibliography

[1] Machine learning glossary. https://developers.google.com/machine-learning/glossary.

[2] A brief history of knowledge graph's main ideas. URL https://knowledgegraph.today/.

[3] *Tree png image with transparent background*. Pngimg.

[4] *WEKA*. The Machine Learning Group at the University of Waikato. URL https://www.cs.waikato.ac.nz/ml/weka/.

[5] Ecotox user guide: Ecotoxicology knowledgebase system. version 5.0. available: http:/www.epa.gov/ecotox/. 1. April 2020.

[6] *History of the software*. Python Software Foundation, 2001. URL https://docs.python.org/2.0/ref/node92.html.

[7] *Apache Spark™is a unified analytics engine for large-scale data processing*. The Apache Software Foundation, 2018. URL https://spark.apache.org/.

[8] *Use Pharmacological Action Terms [mh] with a Drug*. National Library of Medicine, 2018. URL https://www.nlm.nih.gov/bsd/disted/drugs/pas.html.

[9] *The History of Java Technology*. Oracle, 2019. URL https://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html.

[10] *MeSH Intro - Preface*. National Library of Medicine, 2019. URL https://www.nlm.nih.gov/mesh/intro_preface.html#pref_rem.

[11] *MeSH Record Types*. National Library of Medicine, 2019. URL https://www.nlm.nih.gov/mesh/intro_record_types.html.

[12] *Deep Learning for Java*. Eclipse Foundation, 2020. URL https://deeplearning4j.org/.

[13] *Training and Test Sets: Splitting Data*. Google, 2020. URL https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[14] *Java™ Programming Language*. Oracle, 2020. URL https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html.

[15] *Apache Jena*. The Apache Software Foundation, 2020. URL https://jena.apache.org/.

[16] *Keras overview*. Google, 2020. URL https://www.tensorflow.org/guide/keras/overview.

[17] *What is Python? Executive Summary*. Python Software Foundation, 2020. URL https://www.python.org/doc/essays/blurb/.

[18] *Oracle and Sun Microsystems*. Oracle, 2020. URL https://www.oracle.com/sun/.

[19] *org.tensorflow*. Google, 2020. URL https://www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary.

[20] *What is Jena?* The Apache Software Foundation, 2020. URL https://jena.apache.org/about_jena/about.html.

[21] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden,

Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[22] Chittaranjan. Andrade. The P value and statistical significance: Misunderstandings, explanations, challenges, and alternatives. *Indian Journal of Psychological Medicine*, 41(3): 210–215, 2019. doi: 10.4103/IJPSYM.IJPSYM_193_19. URL `http://www.ijpm.info/article.asp?issn=0253-7176;year=2019;volume=41;issue=3;spage=210;epage=215;aulast=Andrade;t=6`.

[23] Tim Berners-Lee and Mark Fischetti. *Weaving the Web; The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper Audio, 1999. ISBN 0694521256.

[24] Carl Boettiger. *rdflib: A high level wrapper around the redland package for common rdf applications*, 2018. URL `https://doi.org/10.5281/zenodo.1098478`.

[25] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf`.

[26] John Cairns, Jr. Ethics in science: Ecotoxicology. *Ethics in Science and Environmental Politics*, 3, 05 2003. doi: 10.3354/esep003033.

[27] François Chollet et al. *Keras*. 2015. URL `https://keras.io`.

[28] Chrislb. File:artificialneuronmodel english.png, 2005. URL `https://commons.wikimedia.org/wiki/File:Neuron_McCullocha-Pittsa.svg`.

[29] Tukur Dahiru. *P–Value, A True Test Of Statistical Significance? A Cautionary Note*. Annals of Ibadan Postgraduate Medicine, 2008. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4111019/.

[30] Alessandro Daniele and Luciano Serafini. Knowledge enhanced neural networks. In Abhaya C. Nayak and Alok Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence*, pages 542–554, Cham, 2019. Springer International Publishing. ISBN 978-3-030-29908-8.

[31] Giuseppe De Giacomo and Maurizio Lenzerini. Tbox and abox reasoning in expressive description logics. volume 1996, pages 37–48, 10 1996.

[32] Parastoo Delgoshaei, Mohammad Heidarinejad, and Mark Austin. Combined ontology-driven and machine learning approach to monitoring of building energy consumption. 10 2018.

[33] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. *Ontology Matching: A Machine Learning Approach*, pages 385–403. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24750-0. doi: 10.1007/978-3-540-24750-0_19. URL https://doi.org/10.1007/978-3-540-24750-0_19.

[34] Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. *Logic Tensor Networks for Semantic Image Interpretation*, volume abs/1705.08968. 2017. URL http://arxiv.org/abs/1705.08968.

[35] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *SEMANTiCS (Posters, Demos, SuCCESS)*, 2016.

[36] The Norwegian Institute for Water Research (NIVA). Niva's computational toxicology program (nctp). 2018. URL https://www.niva.no/en/projectweb/nctp.

[37] The Norwegian Institute for Water Research (NIVA). Ecotoxicology and risk assessment. 2019. URL https://www.niva.no/en/research/ecotoxicology_and_risk_assessment.

[38] Yuxia Geng, Jiaoyan Chen, Zhiquan Ye, Wei Zhang, and Huajun Chen. *Explainable Zero-shot Learning via Attentive Graph Convolutional Network and Knowledge Graphs*. IOS Press, 2019.

[39] Vera Hermine Goebel. *Lecture charts: Knowledge Discovery in Databases (KDD) - Data Mining (DM)*. Department of Informatics, University of Oslo, 2016.

[40] RDF Working Group. *VOCABULARIES*. The World Wide Web Consortium, . URL `https://www.w3.org/standards/semanticweb/ontology`.

[41] RDF Working Group. *Resource Description Framework (RDF)*. The World Wide Web Consortium, . URL `https://www.w3.org/RDF`.

[42] Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. *CoRR*, abs/1702.05563, 2017. URL `http://arxiv.org/abs/1702.05563`.

[43] Mir Henglin, Gillian Stein, Pavel V. Hushcha, Jasper Snoek, Alexander B. Wiltschko, , and Susan Cheng. Machine learning approaches in cardiovascular imaging, 2017. URL `https://www.ahajournals.org/doi/full/10.1161/circimaging.117.005614`.

[44] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard de Melo, Claudio Gutiérrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ArXiv*, abs/2003.02320, 2020.

[45] User:Ignacio Icke. *File:Overfitting.svg*. wikimedia. URL `\url{https://commons.wikimedia.org/wiki/File:Overfitting.svg}`.

[46] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. *CoRR*, abs/1705.10744, 2017. URL `http://arxiv.org/abs/1705.10744`.

[47] Karl G. Kempf, Pınar Keskinocak, and Reha Uzsoy. *Planning Production and Inventories in the Extended Enterprise*. Springer, 2011.

[48] Freddy Lecue. *On The Role of Knowledge Graphs in Explainable AI*. IOS Press, 2019.

[49] Bin Liu, Li Yao, Zheyuan Ding, Junyi Xu, and Junfeng Wu. Combining ontology and reinforcement learning for zero-shot classification. *Knowledge-Based Systems*, 144, 12 2017. doi: 10.1016/j.knosys.2017.12.022.

[50] Per Lytsy. P in the right place: Revisiting the evidential value of p - values. *Journal of Evidence-Based Medicine*, 11, 11 2018. doi: 10.1111/jebm.12319.

[51] Stephen Marsland. *Machine Learning - An Algorithmic Perspective, second edition*. CRC Press, 2015.

[52] Steve Menard and Luis Nell el al. Jpype documentation, 2018. URL `https://jpype.readthedocs.io/en/latest/`.

[53] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. arxiv.org/abs/1301.3781, 2013.

[54] Erik B. Myklebust, Ernesto Jimenez-Ruiz, Jiaoyan Chen, Raoul Wolf, and Knut Erik Tollefsen. Tera: the toxicological effect and risk assessment knowledge graph arxiv:1908.10128. 2019.

[55] Erik B. Myklebust, Ernesto Jimenez-Ruiz, Jiaoyan Chen, Raoul Wolf, and Knut Erik Tollefsen. Knowledge graph embedding for ecotoxicological effect prediction arxiv:1907.01328. 2019.

[56] Francis Sahngun Nahm. *What the P values really tell us, doi: 10.3344/kjp.2017.30.4.241*. Korean Association of Medical Journal Editors, 2017.

[57] Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchý. Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey, doi: 10.1007/s10462-018-09679-z, 2019. URL `https://link.springer.com/content/pdf/10.1007/s10462-018-09679-z.pdf`.

[58] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. *CoRR*, abs/1510.04935, 2015. URL `http://arxiv.org/abs/1510.04935`.

[59] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018. URL `http://arxiv.org/abs/1811.03378`.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] Lutz Prechelt. Early stopping - but when? 03 2000. doi: 10.1007/3-540-49430-8_3.

[63] Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1184. URL `https://www.aclweb.org/anthology/D17-1184`.

[64] Kim S, Chen J, Cheng T, Gindulyte A, He J, He S, Li Q, Shoemaker BA, Thiessen PA, Yu B, Zaslavsky L, Zhang J, and Bolton EE. Pubchem 2019 update: improved access to chemical data, doi:10.1093/nar/gky1033. 2019.

[65] John Salatas. File:multilayer neural network.png, 2011. URL https://commons.wikimedia.org/wiki/File:Multilayer_Neural_Network.png.

[66] Bess Schrader. What's the difference between an ontology and a knowledge graph?, 2020. URL https://enterprise-knowledge.com/whats-the-difference-between-an-ontology-and-a-knowledge-graph/.

[67] Arne Seeliger, Matthias Pfaff, and Helmut Krcmar. Semantic web technologies for explainable machine learning models: A literature review. 10 2019.

[68] Arne Seeliger, Matthias Pfaff, and Helmut Krcmar. Semantic web technologies for explainable machine learning models: A literature review. 10 2019.

[69] Baoxu Shi and Tim Weninger. Open-world knowledge graph completion. *CoRR*, abs/1711.03438, 2017. URL http://arxiv.org/abs/1711.03438.

[70] Jun Shi, Huan Gao, Guilin Qi, and Zhangquan Zhou. Knowledge graph embedding with triple context. pages 2299–2302, 11 2017. doi: 10.1145/3132847.3133119.

[71] Amit Singhal. Introducing the knowledge graph: things, not strings. 2012. URL https://www.blog.google/products/search/introducing-knowledge-graph-things-not/.

[72] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[73] Matt Swain. Pubchempy. 2014. URL https://pubchempy.readthedocs.io/en/latest/.

[74] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

[75] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016. URL http://arxiv.org/abs/1606.06357.

[76] user:Aphex34. File:typical cnn.png, 2015. URL https://en.wikipedia.org/wiki/File:Typical_cnn.png.

[77] user:Marobi1. File:semantic web stack.svg, 2014. URL https://commons.wikimedia.org/wiki/File:Semantic_web_stack.svg.

[78] Frank van Harmelen and Annette ten Teije. A boxology of design patterns for hybrid learning and reasoning systems. *CoRR*, abs/1905.12389, 2019. URL http://arxiv.org/abs/1905.12389.

[79] Christina Voskoglou. What is the best programming language for machine learning?, 2017. URL https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7.

[80] C.H. walker, R.M. Sibly, S.P. Hopkin, and D.B. Peakall. *Principles of Ecotoxicology*. CRC Press, 2012.

[81] Kai Wang, Yu Liu, Xiujuan Xu, and Dan Lin. Knowledge graph embedding with entity neighbors and deep memory network. *CoRR*, abs/1808.03752, 2018. URL http://arxiv.org/abs/1808.03752.

[82] 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*, 29(12):2724–2743, 2017.

[83] Ron Wasserstein. *American Statistical Association Releases Statements On Statistical Significance And P-Values*. American Statistical Association, 2016. URL https://www.amstat.org/asa/files/pdfs/P-ValueStatement.pdf.

[84] Geoffrey I. Webb. *Overfitting*, pages 744–744. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_623. URL https://doi.org/10.1007/978-0-387-30164-8_623.

[85] Leiming Yan, Zheng Yuhui, and Jie Cao. Few-shot learning for short text classification. *Multimedia Tools and Applications*, 77, 02 2018. doi: 10.1007/s11042-018-5772-4.

[86] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Learning multi-relational semantics using neural-embedding models. 2014.

[87] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, corr abs/1412.6575. 2015.

[88] Wenhao Zhang. Machine learning approaches to predicting company bankruptcy. *Journal of Financial Risk Management*, 06:364–374, 01 2017. doi: 10.4236/jfrm.2017.64026.