Knowledge and Learning
Synergies between Ontologies and Machine Learning

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von M.Sc. Martin Glauer

geb. am 01.07.1989 in Magdeburg

Gutachterinnen/Gutachter

Prof. Dr. Till Mossakowski
Prof. Dr. Robert Hoehndorf
Prof. Dr. Janna Hastings

Magdeburg, den 10.10.2024

Otto-von-Guericke University Magdeburg

Department of Computer Science
Institute for Intelligent Cooperating Systems

Doctoral Thesis

Knowledge and Learning
Synergies Between Ontologies and Machine Learning

Author:
Martin Glauer

October 10, 2024

Advisers:

Supervisor
Prof. Dr.-Ing. Till Mossakowski

Department of Computer Science
Otto von Guericke University
Universitätsplatz 2
39106 Magdeburg, Germany

Supervisor
Prof. Dr. Janna Hastings

School of Medicine
University of St.Gallen
St.Jakob-Strasse 21
CH-9000 St.Gallen, Switzerland

Glauer, Martin:
Knowledge and Learning
Synergies Between Ontologies and Machine Learning
Doctoral Thesis, Otto von Guericke University
Magdeburg, 2024.

Contents

Abstract

The growing pool of available information and knowledge makes suitable knowledge representation indispensable. Over the past decades, ontologies have emerged as a suitable method for describing knowledge flexibly and with rich semantics. However, ontology development is a labor-intensive and time-consuming process that requires consensus-building processes involving many experts.

This growth in available information is further enhanced by machine learning methods. These have delivered astonishing results, particularly in recent years. However, many of them suffer from a transparency problem.

In this thesis, I show synergies between the world of ontologies and that of machine learning that can address their respective problems. We will present a novel, machine-learning-based approach to extend ontologies. Existing approaches rely largely on existing domain literature, which does not reflect the consensus that inheres in the ontology. Contrary to that, our approach is based solely on an existing ontology and its annotations. I will evaluate the proposed prediction system on a case study on the ChEBI ontology and demonstrate an interface that can be used by ChEBI developers in order to accelerate the development process.

Conversely, we will also present two methods with which ontologies can be used in machine learning. Firstly, I will show that an ontology can serve as a knowledge base for learning tasks, even if the knowledge relevant to the task is not explicitly represented. Furthermore, I will present a method to use the semantic structure of the ontology to improve the logical consistency of predictions.

Zusammenfassung

Der wachsende Fundus an verfügbaren Informationen und Wissen macht eine gute Wissensrepräsentation unerlässlich. Im Laufe der vergangenen Jahre haben sich Ontologien als geeignete Methoden etabliert, um Wissen flexibel und mit reichhaltiger Semantik zu beschreiben. Das Erstellen dieser Ontologien ist jedoch eine arbeitsintensive und zeitaufwendige Aufgabe, die voraussetzt, dass eine Gruppe von Domänenexperten einen gemeinsamen Konsens findet.

Die Geschwindigkeit, mit der die verfügbaren Informationen anwachsen, hat sich seit der steigenden Popularität von Methoden des maschinellen Lernens stark erhöht. Diese haben zwar beeindruckende Resultate in den verschiedensten Domänen geliefert, leiden jedoch unter ihrer mangelnden Transparenz.

In dieser Arbeit möchte ich die Synergien zwischen der Welt der Ontologien und der des maschinellen Lernens aufzeigen, die diese jeweiligen Probleme adressieren können. Ich werde einen neuartigen Ansatz präsentieren, welcher maschinelles Lernen nutzt, um Ontologien zu erweitern. Existierende Ansätze hierfür beruhen häufig auf Domänenliteratur, welche nicht den Konsens einer bestehenden Ontologie reflektieren. Im Gegensatz hierzu beruht dieser Ansatz ausschließlich auf der Ontologie und den enthaltenen Annotationen. Ich werde das resultierende Vorhersagesystem anhand der ChEBI-Ontologie evaluieren und einen neuen Service vorstellen, mit dem die Entwickelnden ihre Arbeit an der Ontologie beschleunigen können.

Weiterhin möchte ich zwei Wege demonstrieren, auf denen Ontologien Methoden des maschinellen Lernens unterstützen können. Zunächst zeige ich, dass Ontologien als Wissensquelle für komplexe Lernaufgaben dienen können, selbst wenn das Wissen, das für die Aufgabe benötigt wird, nicht explizit in der Ontologie repräsentiert wird. Weiterhin werde ich eine Methode vorstellen, mit der die semantische Struktur der Ontologie direkt in das Training eingebunden werden kann, um konsistentere Vorhersagen zu fördern.

1 Introduction and Motivation

A good understanding and overview of the latest research results have always been the basic prerequisites for successful scientific work. But in recent years, the pace of scientific discovery increased significantly due to the use of computational methods. Artificial intelligence in particular has had a major impact on the speed with which new scientific results are produced and published. While this proliferation of information holds great promise for advancing our understanding of the world, it also poses significant challenges to researchers who aim to preserve their broad expertise within their own domains.

This development is particularly prominent in the field of research data publication. The widespread and accepted standard used to be that it was sufficient to make research data publicly and freely available. However, in the face of the rapidly growing landscape of scientific data, it has become increasingly difficult to find data and to understand the data found. To address these and other problems in the growing data landscape, the FAIR principles [1] were introduced in 2016, which define concise and measurable quality criteria for the publication of reusable, open data. The name is derived as an acronym of the four principles: Findability (F), Accessibility (A), Interoperability (I) and Reuseability (R).

In the course of the work of our group – independent of the work done in this thesis – on the development of the Open Energy Platform[1], a data platform for open energy systems modelling, these criteria were the focus of many design decisions. The platform allows researchers to publish domain-relevant data and annotate it with appropriate metadata in a way that complies with the FAIR principles. However, this development process has shown that a purely superficial description of data records is not sufficient to really fulfil the FAIR principles and it also quickly became clear that a purely text-based annotation was not sufficient here. Like most other scientific communities,

[1] `https://openenergy-platform.org/`

the energy modelling community consists of many small and often insular sub-communities that use their own organically evolved terminologies. These terminologies are often not compatible across communities. As a result, annotations that are correct and understandable in one community may be misinterpreted by members of another community. It is therefore not enough to describe the data superficially.

During our development of the Open Energy Ontology [2], we asked different experts from the domain for a definition for the term "powerplant". The result was a collection of three distinct definitions. For simplicity, we will illustrate these three different concepts using the example of a wind turbine. For one group of experts, a powerplant was a single wind turbine. For another group, a powerplant was a whole wind park. The third group defined a powerplant as just the electrical generator part that is installed inside the turbines top. A naive integration of data sources regarding the energy production of powerplants from these three groups would lead to an inconsistent dataset. Therefore, these underlying ambiguities must be identified and resolved before any data integration can take place. A description of the semantics underlying the data is required.

The problems we mentioned above for the energy system modelling community are of course not limited to them. In fact, these problems have been occupying experts in biology, chemistry and medicine in particular for some time. For centuries, biologists have been studying the processes within living organisms of various kinds, while chemists deal with chemicals and their interactions. Obviously, the processes that interest biologists are based on chemical processes. A cross-domain integration of data sources would therefore benefit research endeavors in both domains. But both communities had developed terminologies and distinction systems that were not compatible with each other and harmonization of concepts from both domains is essential for successful data integration. Similar to the previous example from the energy domain, it should also be clear here that a simple mapping of concepts between the domains is not realistic. At the same time, however, both domains deal with different aspects of the same reality. The semantics behind the relevant concepts must therefore still be compatible. In order to overcome the hurdles of knowledge integration between the domains of biology and chemistry, the experts developed numerous domain-specific semantic technologies under the umbrella of the *Open Biological and Biomedical Ontology Foundry* [3] (short: OBO foundry). As the name indicates, OBO focuses mainly on the development of ontologies for different sub-domains within the bio-chemical nexus. We will introduce ontologies more thoroughly in chapter 2. Broadly speaking, ontologies are a means to describe concepts within a domain and relations between these concepts.

Two of the most successful ontologies developed within this context are the Gene Ontology[4, 5] (GO) and the ontology of Chemicals Of Biological Interest[6, 7, 8] (ChEBI). Both ontologies have been widely adopted within their respective communities and have been used in a variety of different applications. The development process of ChEBI in particular, however, highlights one of the main challenges in modern ontology development. ChEBI Release 229, which was published on 1st January 2024, contains 61,065 fully annotated chemical compounds. Over the past 100 monthly releases, an average of 158 compounds have been added to ChEBI per month [9]. This level of maintenance already requires a considerable amount of effort. Yet, this rate

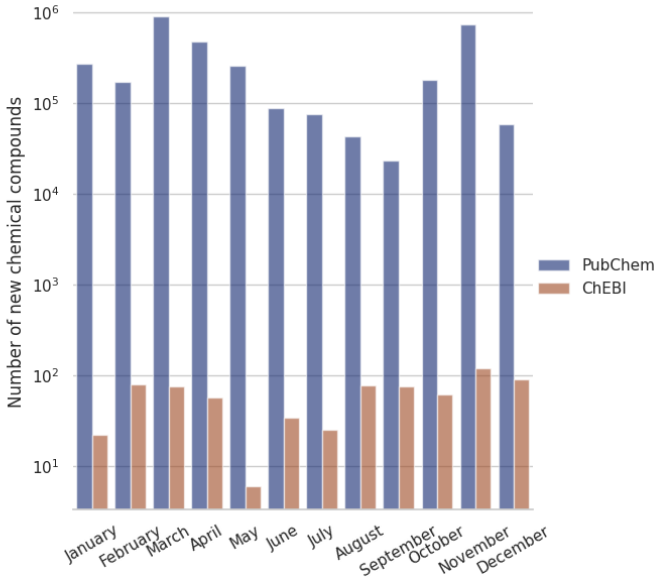


Figure 1.1: Comparison of new compounds added to ChEBI (derived from [9]) and PubChem (derived from [10]) per month in 2023 on a log-scale.

of growth is dwarfed in contrast to the rate of scientific discovery in the field. PubChem [11], a popular but mostly unstructured data source for chemical compounds, features a set of 116,125,060 chemical compounds[2]. Figure 1.1 compares the growth of both resources in 2023. It can be seen that not only has PubChem vastly outperformed ChEBI by several orders of magnitude, but the growth of ChEBI has also been significantly below its own 100-release average of 158. This decline in growth is typical for ontology development, because the addition of new classes requires the consideration of an increasingly large hierarchy and the distinctions on lower hierarchy levels are often more nuanced and hard to define. Ontologies also suffer from similar structural problems as scientific software. After large and urgent problems addressed by

[2] Sourced from `https://pubchem.ncbi.nlm.nih.gov/docs/statistics` on 25th January 2024

ontology have been dealt with, there is often a lack of long-term funding. The focus of researchers, developers and management is shifting to other important areas. Ensuring sustainable maintenance and further development and training new developers then often becomes a sideline for enthusiasts. Yet, ontologies are important resources that are required to structure the complex knowledge in these domains. Therefore, new development techniques and support systems are required to support the ontology development process.

Artificial intelligence methods have been a major driving factor in scientific discovery in recent years. Large language models in particular have been in the focus throughout 2023 and public applications of generative AI models such as ChatGPT have sparked significant public interest. According to Gartner's *"Hype Cycle for Data and Analytics Programs and Practices, 2023"* [12], generative AI models are currently at the peak of their "hype curve". One particular driver that is referenced in this analysis is its ability to support software development, with an estimate that generative AI models "can automate up to 30% of a programmer's work". This raises the question of whether these systems could play a similar role in the creation and evolution of ontologies. Ontology development involves defining concepts, relationships, and axioms to represent knowledge in a structured and formalized manner. If generative AI models can streamline software development by automating coding tasks, it prompts exploration into whether they could assist in generating or refining ontological structures, potentially accelerating the often intricate and time-consuming ontology engineering process. However, the Gartner analysis also covers challenges faced by generative AI models, including the issue of hallucinations. The term "hallucination" refers to the generation of incorrect or nonsensical information by the model. In the context of ontology development, this poses a significant hurdle as inaccuracies could compromise the conceptual and logical integrity of the knowledge representation. Therefore, different techniques are required to play this role.

Gartner's analysis also highlights another problem with popular generative AI models. Most modern AI approaches do not allow detailed insights into their internal workings or the actual knowledge that they represent. Yet, a detailed understanding is crucial for application of these systems in contexts that are critical. An automobile producer that integrates artificial intelligence into their self-driving cars has to guarantee that the car behaves in an expected way even in unexpected or unusual situations. To quote the analysis by Gartner [12], *"Hallucinations, factual errors, bias, a black-box nature [...] preclude the use of generative AI for critical use cases."*

Purely data-driven machine learning approaches learn to extrapolate from a dataset. Without a proper method to understand and evaluate the internal mechanisms, one cannot predict how a model will react when confronted

with previously unseen data. The usual solution for this problem is a larger, more diverse dataset to fill the gaps. However, given a sufficiently complex problem, there will always be a case that has not been covered by the training data. An extension of training data may also not be possible or very expensive. Imagine a model that aims to predict the toxicity of chemicals. Generating new data would require possibly lethal experiments on cell cultures or even live animals. The integration of expert knowledge in these systems can reduce the amount of data needed to reach the desired predictive performance for a model and ontologies can be an excellent source of this knowledge.

1.1 Objectives of this Thesis

In this thesis, we want to present and analyze different ways in which ontology development and machine learning can benefit each other in order to address the problems that we pointed out earlier.

Objective 1: Machine learning aids ontology development Ontology development is often slow and labour intensive. This is primarily due to the rigour required when introducing new concepts into an existing ontology, as these additions must align with a consensus among various experts and communities and also with the general structure of the ontology and with its upper ontology. Ensuring a cohesive representation of this consensus requires consistent design decisions throughout the entire ontology. As part of this first objective, we introduce and analyze a system capable of training on an existing ontology and subsequently expanding upon it. To achieve this, we use specific structural annotations that contain information regarding the intended semantics of a class. We will present multiple approaches to this method and evaluate them using the ChEBI ontology. This ontology contains classes are annotated with string representations of molecules serving as the aforementioned structural annotations.

Objective 2: A support system for ChEBI developers Ontology development is always a collaborative effort that cannot be successful without the active contribution of experts from the domain. These experts are often well-established and active experts in their respective domain and consequently highly sought-after and involved in many valuable efforts outside of ontology development. It is therefore mandatory that a system that supports ontology development is tailored toward the specific needs of these experts and can be used in a meaningful and frictionless way. As part of this objective, we present a web-based framework that aims to enable stakeholders to use the system developed as

part of Objective 1. We will present the different features of this system and discuss the result of a user study and subsequent developments.

Objective 3: Ontologies aid machine learning Ontologies are also a valuable resource for expert knowledge. The considerable effort that must be invested when developing a high-quality domain ontology results in a precise, consistent and well-structured representation of important domain knowledge. This knowledge can also be used as a resource that guides machine-learning approaches. As part of this goal, we will demonstrate how the integration of ontological knowledge into the learning process can impact the predictive quality of a model. Herein, we will focus on two different techniques. First, we will demonstrate the impact of an additional ontology-based pre-training step on tasks that are not directly covered by the ontology. Second, we will present a method to integrate the logical framework that underlies many ontologies to guide the model during training.

1.2 Structure of this Thesis

We would like to address these three objectives in this thesis. To do so, we will first introduce the concepts that are relevant to understanding this work in Chapter 2. In Chapter 4, we will then present a system with which an ontology can be automatically extended. We will introduce the individual building blocks that make up this system and evaluate them using the ChEBI ontology. Furthermore, in this chapter we will present an interface that allows ontology developers to easily interact with our system. In Chapter 5, we will present two different ways in which ontologies can also help with machine learning. Here we will demonstrate how knowledge from an ontology can contribute to better learning of chemical toxicity. Furthermore, we will present a special ontology-based loss function that leads to better consistency in the trained model. Finally, we will summarise the discussed results in Chapter 6 and give an outlook on the future perspectives of the methods we have investigated and how they will be continued in a new research project.

2
Background

The aim of this work is to bring the worlds of machine learning and ontologies closer together. An understanding of basic concepts from both areas is therefore necessary. In this chapter, we will introduce those concepts that are important to understanding the methods proposed in this thesis.

2.1 Ontologies

Ironically, the term "ontology" is a very overloaded one that is interpreted differently in different domains. Ontologies have their origins in philosophy as a "study of being", in which a generalized categorization of existing things and their relationships is sought. Here the concept of ontology reaches back to Aristotle's "Metaphysics" and revolves around the question of the structure of reality - as a whole or in specific parts. Recently, however, the term has been taken up by the field of knowledge representation. This field studies methods to express existing knowledge in ways that are human and machine-readable. When it comes to ontologies, the fundamental question remains the same: "How can certain aspects of reality be captured correctly and consistently?". However, since knowledge representation often deals with specific use cases that are also intended to be used in software solutions, a more pragmatic notion of ontologies than the one studied in pure philosophy is required here.

The most widespread definition of the term ontology comes from Gruber [13] and has since been expanded [14] to

Definition 1 (Ontology [13, 14]) *An ontology is a formal, explicit specification of a shared conceptualization.*

But assuming we were to use this definition to build a system for ontology development. What are the components of an ontology? What kinds of properties are important when building an ontology? Definition 1 does not provide a sufficient framework to answer these questions. This is because this defini-

tion uses terms that themselves require definition [15]. An alternate definition has been proposed by Neuhaus in 2018 [16].

Definition 2 (Ontology [16]) *An ontology of a given domain of interest is a document that provides*

1. *a vocabulary for describing the domain of interest,*
2. *annotations that document the vocabulary, and*
3. *a logical theory (consisting of axioms and definitions) for the vocabulary, in a way that these three elements together enable a competent user of the ontology to ascertain its intended interpretation.*

That work also contains a more formal definition that yields a precise mathematical specification of what an ontology is. For the context of this work, however, this informal definition is sufficient. Compared to the definition by Gruber, it is also closer to the reality of how ontologies for knowledge representation are built in practice. Ontologies play a crucial role in knowledge representation by providing a structured framework for organizing and categorizing information in a specific domain. An ontology defines a set of concepts and their relationships, that capture the essential elements within a particular knowledge domain. These concepts are often formalized using the *Web Ontology Language* (OWL). The OWL language consists of classes (e.g. dog or mammal) and their relations. The most prominent and most frequently used of these relations is the subclass relation. OWL also allows the expression of complex relationships such as "Every dog is a mammal", "A dog is not a cat" or "A dog is a mammal that barks.". These relationships are expressed as logical axioms. These are usually not the most accessible way to describe concepts to non-logicians. OWL therefore also allows for the textual annotation of these classes and relations. High-quality annotations allow domain experts easy access and understanding of the concepts defined in the ontology.

2.2 Chemical Entities of Biological Interest (ChEBI)

Chemical Entities of Biological Interest [8, 7] (ChEBI) is a comprehensive and widely utilized database and ontology that focuses on the representation of molecular entities that occur in biological processes. Developed and maintained by the European Bioinformatics Institute (EBI), more specifically EMBL's *European Bioinformatics Institute* (EMBL-EBI), ChEBI serves as a valuable resource for researchers, scientists, and professionals in the fields of bioinformatics, chemistry, and life sciences.

ChEBI is designed to provide a systematic and structured classification of chemical entities, encompassing a diverse range of molecules such as small molecules, peptides, nucleic acids, carbohydrates, and more. Larger chemical structures such as proteins [17], genes [4, 18] are not part of ChEBI, but covered by different ontologies under the umbrella of the *Open Biological and Biomedical Ontology* foundry (OBO, [19]). The ontology employs a controlled vocabulary to annotate and classify molecular entities, enabling users to explore relationships, similarities, and distinctions between various compounds.

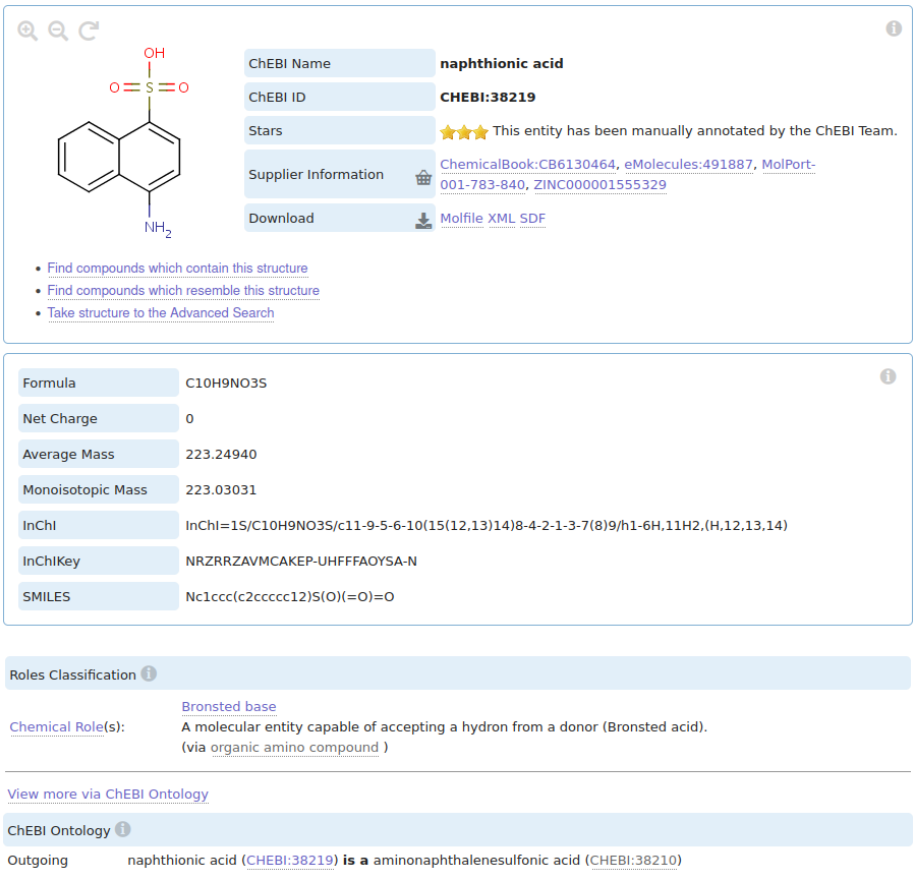


Figure 2.1: Screenshot of the representation of the class *naphthionic acid* (CHEBI:38219).

Figure 2.1 shows the class of *naphthionic acid* (CHEBI:38219) as represented in ChEBI. The molecular structure on the left shows that this class is fully specified - that is, that it is annotated with a specific structure that all instances of this class share. The structure shows the two ortho-fused benzene rings that are characteristic for *naphthalenes* (CHEBI:25477), a superclass of *naphthionic acid* (CHEBI:38219), as well as the sulfo group linked to a car-

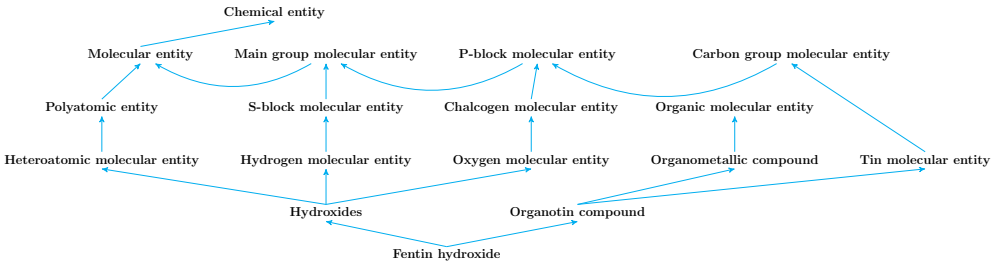


Figure 2.2: *Fentin hydroxide* and its hierarchical classes. Blue lines indicate the *subsumption* relationships.

bon of an aryl group, that classifies this structure as an *arenesulfonic acid* (CHEBI:33555), another superclass of *naphthionic acid* (CHEBI:38219).

The box right of the molecular structure contains fundamental information about this entry, such as its name, its ChEBI-identifier, and information regarding the curation level of this entry, encoded by the number of stars ranging between zero and three. Three stars denote entries that have been manually annotated by ChEBI developers; two stars if an entry has been annotated by a third party. As of Release 229, ChEBI there are 60,882 fully annotated 3-star compounds and 134,778 2-star compounds. In this work, we use release 200, which contains 59,108 3-star compounds and 80,611 2-star compounds. Single-star annotations are mostly preliminary entries that have not yet been manually annotated, while no stars denote entries that are deleted. This release was the most recent version when we started our investigation of the presented methods. To ensure comparability between our results, we fixed this version for our analyses, unless stated otherwise.

Below this box is an overview of important chemical qualities such as the net charge and average mass. Most important for this work is the SMILES annotation. We will explain this string representation of molecular structures in detail in Section 2.3.1. ChEBI also annotates roles such as potential uses of chemicals. In this example, one can see that *naphthionic acid* (CHEBI:38219) can be a Brønsted base. While this role contains information about the chemical's reactive behavior, there are also roles that represent potential uses or biological roles, such as poison (CHEBI:64909) or vitamin (CHEBI:33229). The methods in this work do not yet utilize role information, but an extension of the proposed system is currently ongoing research and will be discussed in Chapter 6.

Lastly, we can see the most important information for this work: The entry's position within ChEBI's taxonomy, i.e. the way this class is connected to others via the subsumption hierarchy. At its core, ChEBI provides a hierarchical classification based on an "is a"-relationship between its classes that can be

very complex as can be seen in Figure 2.2. The part of the taxonomy that we are focused on in this work starts with the root class *"chemical entity"* (CHEBI:24431), which in turn has four important subclasses:

atom (CHEBI:33250) Atoms are the fundamental building blocks of chemical substances. This subclass focuses on these smallest components that constitute matter at the atomic level.

group (CHEBI:24433) Groups are functional or otherwise important substructures within molecules. The *group* subclass delineates the specific components that contribute significantly to the molecular structure's functionality.

molecular entity (CHEBI:23367) : Distinguishable chemical structures comprising atoms, bonds, and other molecular components. Molecules, in this context, represent specific and distinguishable arrangements of chemical elements.

chemical substance (CHEBI:59999) : Compositions of molecular entities, including mixtures, the Substances subclass encapsulates complex combinations of molecules. It extends beyond the individual molecular level to encompass higher-order arrangements of chemical entities.

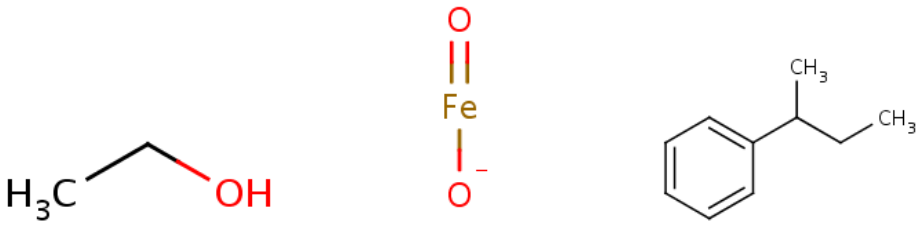
In this work, we are interested in concrete chemical structures. Therefore, the branch of molecular entities is particularly important. It is important to note the intended semantics of a class in ChEBI. Take a bucket of water as an example. The body of water contained in this bucket would not be a member of the class *water* (CHEBI:15377), but each individual molecule therein would. The body of water is an aggregate of several instances and would therefore be classified as a chemical substance - depending on its precise composition, a *pure substance* (CHEBI:60003) or a *mixture* (CHEBI:60004). Therefore, we will not be concerned with actual instances of classes in ChEBI but, instead, put a larger focus on their structural definitions in the form of SMILES representations.

2.3 Cheminformatics

2.3.1 SMILES

The Simplified Molecular-Input Line-Entry System (SMILES) is a formal language that allows a string-based representation of chemical structures. Its original version was published in 1988 [20] as a means to represent and exchange chemical structures in a machine-readable way. Ever since, it has

been one of the widely used representations for chemical structures that is now maintained as a proprietary standard by Daylight Chemical Information Systems, Inc. This proprietary character did, however, hinder the widespread application of SMILES. In 2007, the Blue Obelisk community developed an open-source version of SMILES, the *Open Smiles* specification [21]. In the following, we will give a basic overview of Open Smiles that is required to understand the input encodings and discussions for the proposed system. The main building blocks of chemicals are atoms and their bonds. These also comprise the fundamental parts of SMILES strings.



(a) *ethanol* (ChEBI:16236) `CCO`

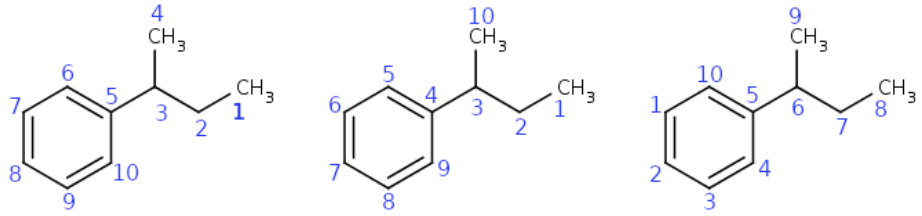
(b) *iron(III) oxide-hydroxide(1-)* (CHEBI:78619) `[O-][Fe]=O`.

(c) *sec-butylbenzene* (CHEBI:35097) `CCC(C)c1ccccc1`.

Figure 2.3: Different molecular structures from ChEBI with their respected SMILES strings.

Each SMILES string represents an atom-wise traversal of the molecular graph of a molecule. Atoms are represented by their elemental symbols (e.g., C for carbon, O for oxygen), and the bonds between them are implied based on their sequential arrangement. For example, in ethanol (CCO), the two carbon atoms are assumed to be connected by a single bond, and the oxygen is attached to one of the carbon atoms. The single bonds are optional and could be made explicit by using a single dash (`-`). Notably, the defining feature of ethanol, namely the hydroxy group bound to a saturated carbon atom is not explicit in its SMILES string. This is the case because in most cases, hydrogen is not explicitly denoted. Instead, SMILES is interpreted under the assumption that each atom has a full valence shell unless explicitly annotated with a charge counter, such as the `[O-]` in Figure 2.3b. The square brackets used in `[O-]` are the default for all atoms, but can be omitted for elements from the organic subset (B, C, N, O, S, P, F, Cl, Br, I). These brackets can contain additional information regarding the enclosed atom, such as isomeres (e.g. `[10C]`), charge (e.g. `[C+]`), connected hydrogen (e.g. `[CH3]`) and chirality (e.g. `[C@@]`). Ring structures as seen in *sec-butylbenzene* in Figure 2.3c, are indicated by numeric position markers (e.g. the six-carbon ring `c1ccccc1`).

The atom preceding each position marker is still part of the ring. In this particular example, the ring is also aromatic, which is indicated by the use of lower-case letters. The structure of *sec-butylbenzene* (CHEBI:35097) also contains a branch that is enclosed in parenthesis. The order in which branches and cycles are processed is not determined by the smiles standard. Figure 2.4



(a) `CCC(C)c1ccccc1` (b) `CCC(c1ccccc1)C` (c) `c1cccc(C(C)CC)c1`

Figure 2.4: Different SMILES representations for *sec-butylbenzene* (CHEBI:35097). Blue numbers indicate the order in which nodes have been visited.

shows different SMILES representations for the same molecule. The choice of the starting atom heavily influences the resulting string. It can be seen in Figure 2.4c that the numeric markers that indicate the beginning and end of the aromatic ring have been separated to the beginning and the end of the entire string.

2.3.2 Molecular Fingerprints

The SMILES language provides a useful framework, but a large number of SMILES structures can encode the same chemical structure. In fact, every node in the molecular graph can be used as the start of the iterative generation process. It is therefore hard to determine whether two SMILES strings actually encode the same chemical structure. At the same time, SMILES strings do not directly encode important chemical features. Due to the way cycles are broken during the graph traversal, these features may be distributed over different parts of the SMILES string. This may cause structures that are close in the molecule, to be far apart in the molecule's SMILES representation. These properties can be detrimental to different machine-learning approaches.

Morgan Fingerprints Morgan fingerprints, also known as Extended Connectivity Fingerprints (ECFP, [22]), aim to provide a fixed-length representation of a molecule that contains important structural features. They are generated by traversing the local environment around each atom in a four-step process. First, each atom in the molecule is assigned a unique integer identifier. These

identifiers are generated using a hash based on the *Daylight atomic invariants rule* [20], which includes the number of non-hydrogen neighbors, number of attached hydrogen, the valency excluding hydrogen bonds, and characteristics of the represented atom (number, mass and charge). The fingerprints used in this work also include a boolean identifier that represents whether the atom is part of a bond.

In the next step, these identifiers are updated in an iterative process that aggregates immediate non-hydrogen neighbors together with their respective bond information. These aggregates are then hashed again into a structure that contains information about the atom itself and its immediate neighborhood. Repeating this process increases the radius of the neighborhood that is covered by this representation such that, after k iterations, each node contains information about its k-neighborhood. The bonds in this neighborhood are also explicitly tracked and hashes that represent the same neighborhood as an already existing one are discarded. This process is repeated until a fixed number of iterations have been performed. Finally, the remaining hashes are turned into fingerprints of fixed length n. These are initialized as a boolean vector. For each hash, the remainder of the division by n is calculated and the corresponding bit in the fingerprint is set to 1.

This generation process is inherently lossy. The aggregation of fingerprints in the final step does not account for bit collisions, i.e. the case in which two hashes have the same remainder. There are, therefore variants of this approach that do not return boolean vectors, but integer vectors that do not lose this information. This special kind of fingerprint is also referred to as the Extended Connectivity Fingerprint Count (ECFC, [23]).

Topological Fingerprints Another frequently used kind of fingerprint is the topological fingerprint, particularly their implementation in the RDKit library [24]: *RDKit Fingerprints*. Like the Morgan fingerprints, these are based on consecutively aggregated hash values. However, the way in which these hashes are generated differs significantly. Instead of being based on individual nodes and their neighborhood, topological fingerprints are calculated on all substructures of a certain size. I.e. all subgraphs containing at most a given number of chemical bonds are aggregated. Alternatively, only all paths of a certain length can be aggregated here. A hash is then calculated for the individual bonds in these substructures, which are then combined to form a hash for the entire graph. These hashes are then also used with a random number generator to set bits in the fingerprint.

2.3.3 ClassyFire

We explained in the introduction that ontology development is a slow, labor-intensive process and that appropriate tool support is needed to speed it up. ChEBI developers already use a tool for classifying chemicals: ClassyFire [25]. This tool was developed in conjunction with the chemical ontology ChemOnt[1]. Contrary to ChEBI, ChemOnt does not use different orthogonal concepts for class distinctions such as structure, function and chemical derivations. Instead, ChemOnt is based solely on structural distinctions. According to the related publication, it contained at the time of publishing a total of 4825 chemical categories – for comparison, the current version of ChEBI contains 61,065 fully annotated compounds.

The ClassyFire tool was then developed based on these structural definitions as a system of classification rules. The vast majority of these manually created rules use the SMARTS notation, an extension of SMILES for molecular pattern matching. More than 9000 SMARTS strings were created. However, the formal description of chemical structures requires a very expressive logic [26]. Therefore, some structures cannot be expressed in SMARTS. Markush structures, a formalism commonly used in the description of chemical structures in patents, have been used for some, while others rely on a logic formalism that allows the arithmetic analysis of ring and atom counts. The exact nature of this formalism remains unspecified in [25].

Some of the classes in ChemOnt can be found in ChEBI, such as its main distinctions 'organic compounds' and 'inorganic compounds', which correspond to the ChEBI classes *organic molecular entity* (CHEBI:50860) and *inorganic molecular entity* (CHEBI:24835). Other classes do not align that smoothly. The class *alkali metal salts* is contained in both ChemOnt and ChEBI as is shown in Figures 2.5 and Figures 2.6. But in ChemOnt, this class is a subclass of *inorganic compound*, while *organic sodium salt* (CHEBI:38700) is a subclass of *alkali metal salt* (CHEBI:35479). The intended semantics of *alkali metal salts* must therefore be different in both ontologies.

During the development of ClassyFire, some mappings to other resources from the domain of chemistry, such as ChEBI, have been created. The resulting look-up table for ChEBI maps ChemOnt classes to an average of 1.24 ChEBI classes. These mappings have also been integrated into the development process. ClassyFire is actively used as a tool by ChEBI's development team in order to propose preliminary classifications for new compounds. This process is limited to classes that are covered by ChemOnt, which is significantly smaller than ChEBI. Consequently, this automated classification will result in classifications that are higher in ChEBI's taxonomy. This process may

[1] `http://classyfire.wishartlab.com/downloads`

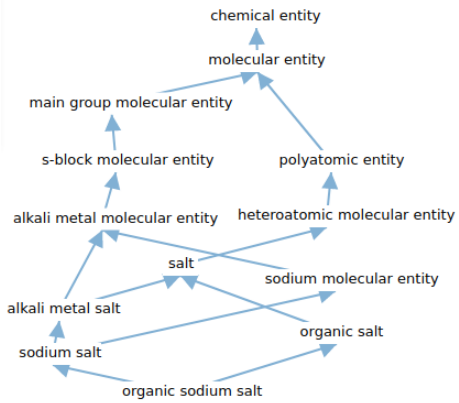


Figure 2.5: Parent classes of *organic sodium salt* (CHEBI:38700) in ChEBI

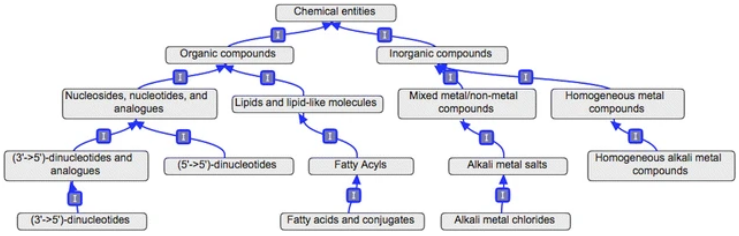


Figure 2.6: Superclasses in ChemOnt as depicted in [25]

affect the structure of future ChEBI versions. Our analysis of ChEBI has shown that certain classes in ChEBI have a significantly larger number of direct subclasses than most others. In ChEBI version 239, there were over 1000 classes that were a direct subclass of *peptide* (CHEBI:16670), which may likely stem from automated classification tools, likely ClassyFire.

This highlights also one of the problems with hand-crafted rules such as the ones employed by ClassyFire. A newer version of the ontology that the tool has been built upon requires a manual extension of the tool. Predictions based on tools that were built against a particular ontology version can only predict classes that were represented in this version - even if more specific classifications would be correct. One possible solution would be to convince the ontology developers to not only produce verbal definitions of ontology classes but also formal ones. A precise formal definition of chemicals is not trivial [26] and the ClassyFire tool itself is based on multiple languages to capture the complexity that inheres in the domain of chemistry. Such a requirement would therefore be likely to further hinder the already slow ontology design process. Furthermore, the ChemOnt ontology has not been maintained since its release 2.1 in 2016[2]. There is a new related ontology, the Chemical Functional Ontology (ChemFOnt, [27]), but at the time of writing, ClassyFire remains in its original version 1.0. This rigidity stifles the usability of tools such as ClassyFire for the development of actively developed ontologies such as ChEBI. A tool that is more adaptable to changes in the underlying ontology would be more desirable.

2.4 Machine Learning

Definition 3 *Classification problems An m-label prediction model is a parameterized function $a_\theta : \mathbb{R}^n \rightarrow [0,1]^m$ with parameters θ from a given parameter space Ω that maps data points from an n-dimensional feature space to m fuzzy class predictions between 0 and 1. Given a partial cost or loss function $c : \Omega \times \{X | X \subseteq (\mathbb{R}^n \times \{0,1\}^m)\} \rightarrow \mathbb{R}$ and a set of instances with corresponding label assignements $P \subseteq \mathbb{R}^n \times \{0,1\}^m$, an m-label classification problem is the optimization problem*

$$\min_\theta c(\theta, P)$$

A (single-label) multi-class classification problem denotes the special case, in which for each instance, exactly one label is positive, i.e. $\forall (x,y) \in P : \sum_{i=1}^{m} y_i = 1$.

[2] `http://classyfire.wishartlab.com/downloads`

Binary classifiers can only distinguish between two disjoint classes ($m = 2$). Despite this limitation, it is still possible to use these classifiers for multi-class problems by using the "one-VS-rest" method. In this method, m different classifiers are trained in which a single class from the original classification problem is compared against an artificial class that represents the union of all remaining classes. The results are m "one-VS-rest" predictions and the one with the highest predicted membership is used as the class prediction. Binary classifiers usually assume that a data point belongs to either class. Therefore, we will simplify our notation for the prediction function and shorten a prediction $a_\theta(x) = (p, 1-p)$ to $a_\theta(x) = p$.

2.4.1 Logistic Regression

In its methodology, logistic regression is closely related to linear binary classification. The predictor is based on a linear hyper-plane $g_\theta(x) : \theta_{n+1} + \sum_{i=1}^{n} \theta_i \cdot x_i$ that separates the two classes:

$$a_\theta(x) = \begin{cases} 1 & g_\theta(x) > 0 \\ 0 & \text{otherwise} \end{cases}$$

In real-world applications, the input data is often noisy, which makes such a crisp classification less meaningful. Instead, the system should express a lower certainty for those points that are close to the separating line. Logistic regression applies an additional sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ to its prediction function:

$$a_\theta(x) = \sigma(g_\theta(x))$$

A commonly used loss function for logistic regression is the mean of the cross-entropy:

$$c(\theta, P) = -\frac{1}{|P|} \sum_{(x,y)\in P} (y_1 \cdot log(a_\theta(x)_1) + y_2 \cdot log(a_\theta(x)_2))$$

2.4.2 K-Nearest Neighbor

The k-nearest neighbor algorithm is one of the few classification algorithms that does not feature immediately trainable parameters. Instead, the labeled data points are directly used for classification based on a given distance metric. Given an unseen datapoint x, this method selects k data points from the labeled dataset that are closest to x. The predicted class is then assigned to that class that represents the majority amongst these k points.

There are different approaches to parameterize this classification method, such as Large Margin Nearest Neighbor that allow it to adapt to different distributions of classes in the feature space.

2.4.3 Decision Trees and Random Forests

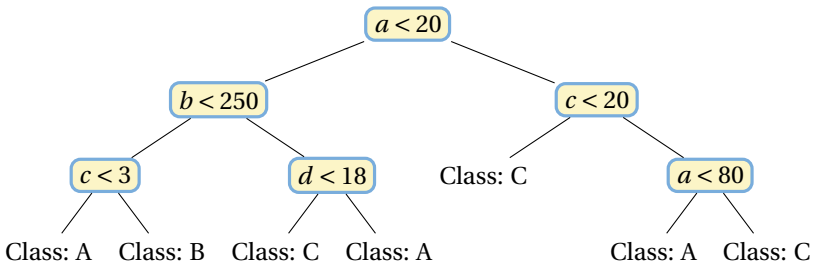


Figure 2.7: Example of a decision tree. Left-hand branches represent a positive answer to the condition in the parent node, right-hand branches represent a negative answer.

Decision trees are classifiers that build a tree structure that can be seen as flow charts and are used for multi-class problems. Each node in the tree represents a decision point and depending on the outcome of this decision, a different branch is taken. Figure 2.7 shows an example of a decision tree. For a given data point $(a, b, c) = (25, 40, 99)$, the processing inside the tree starts at the root node. The root node has the attached expression $a < 20$, which is not satisfied by the given datapoint ($a = 25$). Therefore, the algorithm follows the right-hand branch. The corresponding next decision criterion $c < 20$ will also be answered negatively and the next right path is taken. Finally, a positive decision is made in the third level for $a < 80$. As the left-hand path is now followed here, we reach the final decision of the decision tree and classify the data point as a member of class A.

In this work, we focus on Classification and Regression Trees (CART). These trees are built based on a recursive process. Given a dataset, the decision criterion for the root node is determined based on a splitting criterion that estimates the 'impurity' within a dataset. The goal is to find a decision that splits the dataset in a way that minimizes the probability that assigning a random class to a given data point relative to the relative frequency $p_i = \frac{|\{(x,y) \in P | y_i = 1\}|}{|P|}$ ($i = 1, ..., m$) would lead to misclassification. The probability for this impurity, also known as the Gini impurity, is

$$I_{\text{Gini}}(p) = 1 - \sum_{i=1}^{m} p_i^2$$

Note that $I_{\text{Gini}}(p) = 0$ if and only if $p_i = 1$ for some class i and, consequently $p_j = 0$ for all others. This means that the Gini Impurity is minimal on a dataset

that contains effectively only a single class - also called a 'pure' dataset. The CART algorithm uses this index by attempting to split the dataset w.r.t. each feature separately. For a simple boolean feature, this splits the dataset into two subsets: Those instances for which the feature is present and the ones where it is absent. The feature that separates the dataset in a way that yields the smallest Gini Impurity is selected. This process is repeated with both subsets until all datasets are pure or no further splits are possible. The result of this process is a binary tree of decision points. In order to classify a new data point, one simply has to traverse these decision points until a leaf node is reached. In the case of a pure leaf node, the assigned class is the one that is pure in the respective subset. In the case of an impure leaf node, the class that has the highest number of members in this leaf is assigned.

The biggest advantage of decision trees is the fact that they are easily explainable. As long as features have fixed meanings, the path of decision nodes that leads to a specific decision also contains the explanation for this decision. For larger data and feature sets, decision trees do, however, tend to become complex which limits their explainability and also often leads to over-fitting.

A random forest [28, 29] is an ensemble of multiple decision trees that are built on sub-samples of the original dataset. A majority vote amongst the classifications derived from all trees is used to determine the predicted class for a given data point.

2.4.4 Naive Bayes

A Naive Bayes classifier is a binary classifier that bases its predictions on a probability distribution. Given certain features x, it predicts the binary class membership based on the conditional probabilities

$$a_\theta(x) = (P(C_1|x_1,\cdots,x_n), P(C_2|x_1,\cdots,x_n))$$

Using bayes theorem, these conditional probablilities for $C \in \{C_1, C_2\}$ can be expressed as $P(C|x_1,\cdots,x_n) = \frac{P(C)P(x_1,\cdots,x_n|C)}{P(x_1,\cdots,x_n)}$. The prior $P(C)$ can either be derived from the given dataset or set by existing expert knowledge. The remaining probabilities cannot be easily derived in real-world applications and are therefore approximated under certain assumptions. First, it is assumed that the dataset is sampled uniformly from the feature space and the denominator $P(x_1,\cdots,x_n)$ is therefore assumed to be constant. The second assumption is more restrictive. In order to compute the conditional probability $P(x_1,\cdots,x_n|C)$ for $C \in \{C_1, C_2\}$ of a feature combination given a class $C \in \{C_1, C_2\}$, one would already require knowledge on how the class one aims to predict is distributed in the features space. This would be akin to a probabilistic, feature-based specification of each class, which is usually not

available. Therefore, the Naive Bayes approach assumes that all features are mutually independent. Under this assumption, the conditional probability of the feature combination collapes to the product of the conditional probabilities of each individual feature:

$$P(C|x_1,\cdots,x_n) \quad \propto \quad P(C)\cdot\prod_{i=1}^{n} P(x_i|C)$$

2.4.5 Linear Discriminant Analysis

One of the main drawbacks of Naive Bayes methods is their independence assumption. Linear Discriminant Analysis follows a similar mechanism but accounts for possible co-variances among features, i.e. it estimates the conditional probability $P(x_1,\cdots,x_n|C)$ without the additional assumption of mutually independent features. Instead, LDA classifiers assume that these probability distributions follow normal distributions with equal covariance $\Sigma_{C_1} = \Sigma_{C_2} = \Sigma$. The conditional probability of a point can then be expressed

$$P(\underbrace{x_1,\cdots,x_n}_{=x}|C) \quad = \quad -\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_C)^T\Sigma^{-1}(x-\mu_C)}$$

As with Naive Bayes, we are only interested in the relation between each class-wise probability and can therefore ignore the constant denominator in front of this expression. Similarly, we can exploit the monotonicity of the natural logarithm and instead consider the log-likelihood

$$\ln(P(x_1,\cdots,x_n|C)) \quad \propto \quad -\frac{1}{2}(x-\mu_C)^T\Sigma^{-1}(x-\mu_C)$$

Therefore, we can estimate the log likelihood of the predictor as

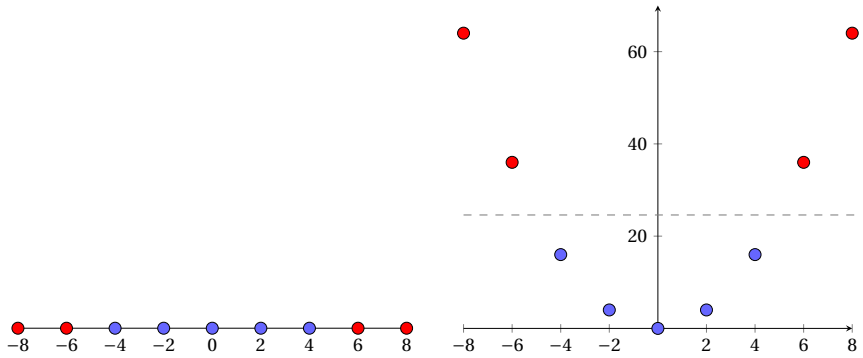
$$\ln(P(C|x_1,\cdots,x_n)) \quad \propto \quad \ln(P(C)) - \frac{1}{2}(x-\mu_C)^T\Sigma^{-1}(x-\mu_C)$$

2.4.6 Support Vector Machine

Support Vector Machines aim to find a hyperplane that separates two classes, and therefore uses the same prediction function as a linear binary classifier. They have, however, the additional goal of keeping the maximal possible margin to the closest point of each class:

$$c(\theta,P) = \|(\theta_1,\cdots,\theta_n)\|^2 + \frac{1}{|P|}\sum_{(x,y)\in P}\max\left(0,(y_2-y_1)\cdot(\theta_{n+1}+\sum_{j=1}^{n}\theta_j\cdot x_j)\right)$$

There are, however, many scenarios in which a linear separation is not possible. In order to circumvent this limitation, Support Vector Machines apply



(a) No kernel (b) Kernel $\phi(x) = (x, x^2)$

Figure 2.8: Comparison of a point separation task with and without kernel.

a non-linear transformation ϕ that maps the feature space into a possibly higher or even infinite-dimensional space. These higher-dimensional spaces allow for a more versatile classification method, but can also be detrimental to the performance of a classifier. In the case of infinite-dimensional transformations, it is not even possible to implement the actual representation of the mapped features. However, it has been shown that the actual explicit representation of this feature space is not necessary. Instead, only the dot product $\langle \phi(x_i), \phi(x) \rangle$ between a given point x and all points x_i (for $(x_i, y_i) \in P$) is required to represent this mapping in a support vector machine. Intuitively, this relation expresses how similar x is to any existing data point. Using Mercer's Theorem [30], we know that the inner product is $\langle \phi(x_i), \phi(x) \rangle$ can be expressed as a positive-semidefinite kernel function $K(x_i, x)$.

This so-called "kernel trick" allows for a simple transformation of the feature space and, thereby, a classifier that can be flexibly adapted to different data distributions. Figure 2.8 shows an example of such a kernel method. The points on the left are not separable by a single linear classifier. By introducing an additional dimension based on a kernel function, they become separable. An example of a separating classifier is given by the dashed line. The choice of the appropriate kernel function, however, is manual and is one of the main deciding factors on whether Support Vector Machines can be successfully applied or not.

2.4.7 LSTM

Traditional neural networks are based on input vectors of fixed size. Many use cases, however, revolve around more flexible data. Natural language, for example, is based on a sequence of words or phones. In order to apply these models to this kind of use case requires elaborate pre-processing, such as fin-

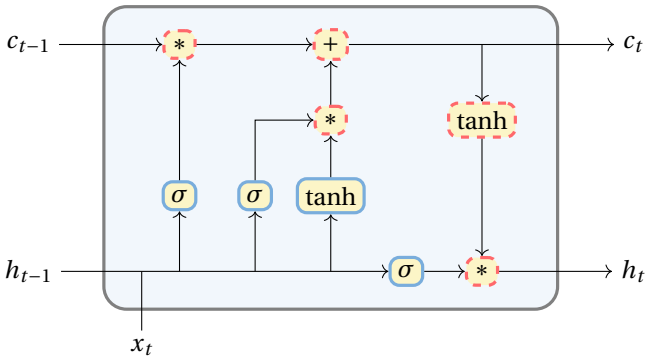


Figure 2.9: Architecture of an LSTM cell. Boxes with blue, solid borders depict dense layers with the respective output functions. Boxes with red, dashes borders depict element-wise operations.

gerprinting or other means of feature extraction are required. Long short-term memories (LSTM, [31]) were proposed as a neural network that is able to process sequences of inputs that represent parts of the input sequence. In natural language processing, these smallest parts - also known as tokens - can be, for example, individual letters. While processing a sequence of tokens, LSTMs maintain an internal state h and a memory c. Figure 2.9 shows a schematic representation of the way an input vector x is processed to update the internal state and memory. Each input vector is processed in sequence. A concatenation of the input vector x and the internal state from the previous step h_{t-1} is passed to three different units, also called gates, that handle the interaction with the memory. The first *forget gate*, uses its input in order to remove parts from the current memory. Next, new values are committed to the memory by the *input gate*. Lastly, the *output gate* produces the next hidden state, which is also the output, based on a combination of its input and the current memory state. The resulting hidden state h_t and memory c_t are then used in order to process the next input vector.

2.4.8 Transformers and Electra

The undoubtedly most prominent architectures in recent years have been Large Language Models (LLM). These systems can process and produce complex natural language and have shown impressive performance in a variety of tasks. Underpinning these architectures is a transformer model. These complex deep neural networks consist of a complex stack of layers of different kinds of networks [32] as depicted in Figure 2.10.

As we discussed before, LSTMs process sequences of input tokens. Transformers follow a similar approach but process the whole input series at once. In or-

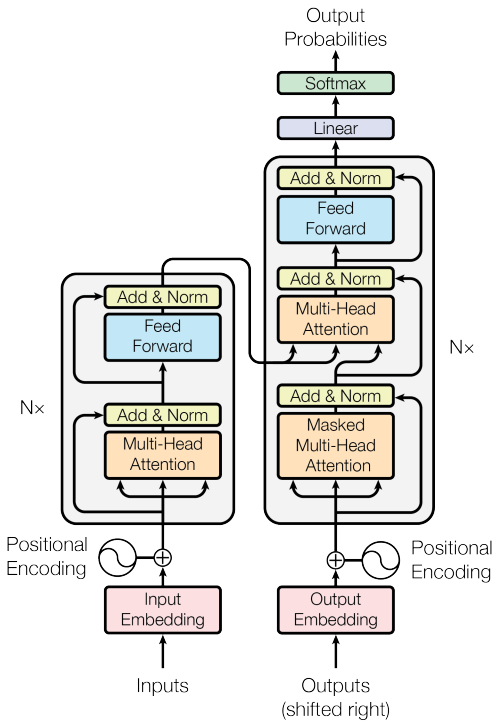


Figure 2.10: Transformer as depicted in [32]

der to still retain information about the order of tokens in the sequence, transformers use positional encodings. These encodings are small offsets that are added to the embedded tokens in order to represent their relative and absolute position within the input sequence. A sinus-based embedding is used for even dimensions and a cosine-based for odd dimensions:

$$\begin{aligned} p(k,2i) &= \sin\left(k\alpha^{-\frac{2i}{d}}\right) \\ p(k,2i+1) &= \cos\left(k\alpha^{-\frac{2i}{d}}\right) \end{aligned}$$

where $p(k,j)$ is the positional encoding for position k at dimension j in an input vector of length d. The user-defined hyperparameter α is fixed to 10,000.

The main architecture of a transformer consists of two parts: An encoder and a decoder. The decoder of a transformer is only used in sequence-to-sequence tasks. In this work, however, we will focus on classification tasks, which are purely based on the encoder half of the transformer model. The left half of Figure 2.10 depicts only a single module of the encoder stack. Each encoder module consists of an attention mechanism and a standard feed-forward layer.

The attention mechanism is one of the central components of a transformer architecture. The intuition behind attention mechanisms is that of a dictionary. Given are tensors of s keys $K \in \mathbb{R}^{s \times d_k}$ and values $V \in \mathbb{R}^{s \times d_v}$ and a query

tensor $Q \in \mathbb{R}^{l \times d_q}$ of length l. In order to find those positions within the key tensor that match the query tensor best, the dot product between their individual vectors is used. The dot product can be seen as a similarity measure between vectors

$$\langle v, w \rangle = |v| \cdot |w| \cdot \cos \sphericalangle(v, w)$$

that is maximal with $|v| \cdot |w|$ for vectors that point in the same direction, 0 for orthogonal vectors and minimal with $-|v| \cdot |w|$ for vectors that point in opposite directions. If an entry at position i, j is particularly large, it means that the query vector $q_{i,\cdot}$ is similar to $k_{j,\cdot}$. In order to represent this relationship as a binary relationship, the softmax of this dot product is computed:

$$\text{Sim}(Q, K) := \text{Softmax}\left(\frac{Q^T K}{\sqrt{d_k}}\right)$$

where $\text{Softmax}(x)$ is the normalization of the exponential of its input vector.

$$\text{Softmax}(x)_i := \frac{e^{x_i}}{\sum_{j=0}^{n} e^{x_j}}$$

The similarity is then applied to v to aggregate those values, whose corresponding keys matched the query:

$$\text{Attention}(Q, K, V) = \text{Sim}(Q, K) \cdot V$$

It can be seen in Figure 2.10 that the input arrow towards the attention box in the encoder block splits into three. This is because the encoder part of the transformer model uses self-attention in which the same input tensor is used for roles of queries, keys and values. Instead, the system is learning three role-dependent projections (M_q, M_k, M_v) and the self attention is then computed as

$$\text{SelfAttention}(x) = \text{Attention}(x \cdot M_q, x \cdot M_k, x \cdot M_v).$$

The calculation of the dot product over the whole dimensionality of the latent space can be costly. Transformers do therefore employ a strategy in which the input tensors are projected to a lower dimensional space, in which the attention mechanism is then computed. These projections are then called attention heads. A transformer can have k attentions heads where k is a divisor of the dimensionality d_{model} of the latent space. Each attention head h then contains its own, independent set of projections $M_q, M_k, M_v \in \mathbb{R}^{d_{\text{model}} \times \frac{d_{\text{model}}}{k}}$. The results of each of these heads are then concatenated in order to regain the full dimensionality of the latent space.

The result of these attentions is then combined with a residual connection, i.e. the original input, as indicated by the curved arrow around the attention

box. The resulting tensor is then normalized using Layer Normalisation [33] that aims to fix the variance of inputs in order to produce more stable results. What follows is a standard feed-forward network followed by another residual connection with Layer Normalisation.

This mechanism can be used to gain some insights into the way information is spread throughout the network. Analyses of the binary relation that is induced by $\mathrm{Sim}(Q,K)$, have been used as means to gain basic interpretability for model predictions [34].

Pre-Training Imagine a model whose task is to classify the messages from a chat according to their sentiment. In order to fulfill this task, the system must first learn a large number of complex subtasks. Even to correctly analyze a simple sentence such as "I don't like that your dog barks so often", the words must be understood in their context. The system must therefore learn the connection between words at the same time it learns to predict sentiments. However, the actual meaning of the words is not necessarily required to establish the connection between these words. Given a sufficient amount of text, it is quite plausible that a model can recognize that in a masked sentence "I don't like that your [MASK] barks so often", the word that was masked was probably "dog". But in order to derive this answer, the model needs to learn the interdependencies of words in a sentence. The task of sentiment analysis that requires often manually labeled data can therefore be separated from the task of learning syntactic connections between words, which does not require manually annotated data.

This procedure is known as masked-language modelling [35], in which unlabeled data can be used to allow the model to learn interdependencies within its input data. A large set of data that consists of inputs is taken and a single element is masked by a fixed token. The model is then trained to correctly predict the token that has been masked. This allows for the training of general-purpose models for a larger domain that can be fine-tuned to address specific tasks within that domain.

The Electra model [36] is essentially a transformer model that uses a variant of pre-training via masked language modeling. This variant co-trains two transformer models. The generator is trained based on the common masked language task. As depicted in Figure 2.11 masked token is then replaced by the token that the generator predicted as the most likely candidate. This altered sequence is then passed to the discriminator, which must predict which token in this sequence was the one that was masked in the first step.

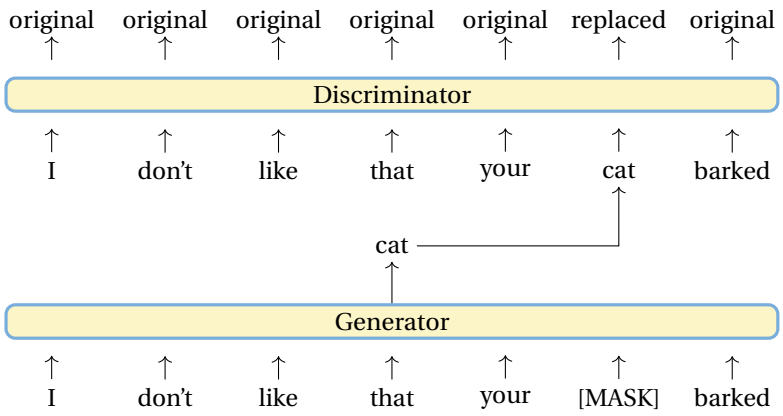


Figure 2.11: The Electra pre-training method. The discriminator wrongfully predicts the masked token to be "cat", allowing the discriminator to detect the replaced token.

2.4.9 Binary Cross-Entropy Loss (BCE)

In Definition 3, we introduced the notion of a classification model over a domain R based on a labeled dataset P, and a model $a_\theta : \mathbb{R}^n \to [0,1]^m$. For a given class i and a datapoint $(x, y) \in P$, the respective label y_i is either 0 or 1. Likewise, the predicted membership $a_\theta(x)_i$ falls into the interval $[0,1]$. Both can therefore be considered to be probability distributions $y_i(x)$ and $a_\theta(x)_i$. The goal when training a classifier is to adapt its parameters θ in such a way that the probability distribution induced by a_θ matches that induced by the ground truth $y(x)$ for all classes. The binary cross-entropy allows us to measure this quality as

$$H(x,y) = -\sum_{i=0}^{n} y_i \log(a_\theta(x)_i) + (1-y_i)\log(1-a_\theta(x)_i).$$

Many deep learning models derive their final predictions using a sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. The binary cross-entropy takes the logarithm of the derived predictions. It can be seen that these operations create the logarithm of a sum of exponentials:

$$\log(\sigma(x)) = \log\left(\frac{1}{1+e^{-x}}\right) = -\log(1+e^{-x}) = -\log(e^0 + e^{-x})$$

This operation is a special case of a function known as the log-sum-exp function $\mathrm{lse}(x_1,...,x_m) = \log\left(\sum_i^m e^{x_i}\right)$. Calculating the gradients for this function is more numerically stable than calculating it for the original operations separately [37].

2.4.10 Evaluation Metrics

2.4.10.1 F1

In order to assess the quality of trained classifiers, one analyses their predictions in relation to the expected outcome. In the case of classifiers that return non-crisp predictions within the interval $[0,1]$, we consider a class prediction as positive, if the predicted value is above 0.5. Given a feature-label pair $(x, y) \in P$ and its prediction $a_\theta(x)_i$ for class C_i there are four possible scenarios as shown in Table 2.1. Let

$$n_{\mathrm{TP},i} := |\{(x, y) \in P | a_\theta(x)_i \geq 0.5 \wedge y_i = 1\}|$$

be defined as the class-wise number of true positive predictions in P for a class C_i. We analogously define $n_{\mathrm{TN},i}$ for true negatives, $n_{\mathrm{FP},i}$ for false positives and $n_{\mathrm{FN},i}$ for false negatives.

	$a_\theta(x)_i < 0.5$	$a_\theta(x)_i \geq 0.5$
$y_i = 0$	True negative	False positive
$y_i = 1$	False negative	True positive

Table 2.1: Possible outcomes given a feature-label pair $(x, y) \in P$ and its prediction $a_\theta(x)_i$ for class C_i with a threshold of 0.5.

Based on these counts, we can define the usual prediction metrics. *Precision* is the probability that a given datapoint that a datapoint that has been predicted as a member of a class indeed belongs to that class:

$$P(y_i = 1 | a_\theta(x)_i \geq 0.5) = \frac{n_{\mathrm{TP},i}}{n_{\mathrm{TP},i} + n_{\mathrm{FP},i}}$$

and *recall* is defined the reversed conditional, i.e. that a given datapoint that is a member of a class is classified as such

$$P(a_\theta(x)_i \geq 0.5 | y_i = 1) = \frac{n_{\mathrm{TP},i}}{n_{\mathrm{TP},i} + n_{\mathrm{FN},i}}.$$

These metrics give a good insight into the general predictive quality of a classifier. However, it is important that both metrics cannot be meaningfully evaluated in isolation. A classifier that never predicts any false positives would result in perfect precision – regardless of how many true positive classifications there actually are. Likewise, a classifier without false negative prediction would result in perfect recall. Both metrics must therefore be considered in

relation to each other. The F1 score represents such a relation by calculating the harmonic mean of the precision and the recall, i.e.

$$\begin{aligned} F_{1,\text{macro}} &:= \frac{1}{m}\sum_{i=1}^{m} \frac{2}{P(y_i = 1|a_\theta(x)_i \geq 0.5)^{-1} + P(a_\theta(x)_i \geq 0.5|y_i = 1)^{-1}} \\ &= \frac{1}{m}\sum_{i=1}^{m} \frac{2n_{\text{TP},i}}{2n_{\text{TP},i} + n_{\text{FP},i} + n_{\text{FN},i}} \end{aligned}$$

Note that all of the above metrics for multi-class classification are calculated class-wise and then averaged, which is called the *macro*-aggregation of a metric. Further, we will also use the *micro*-aggregation, which aggregrates true positives, false positives and false negatives globally before computing the F1-measure:

$$F_{1,\text{micro}} := \frac{\sum_{i=0}^{m} 2n_{\text{TP},i}}{\sum_{i=0}^{m} (2n_{\text{TP},i} + n_{\text{FP},i} + n_{\text{FN},i})}$$

While the *micro*-aggregation yields a general overview of the predictive performance, it does not work as well on an imbalanced dataset. Consider a dataset that consists of two classes A and B. The dataset consists of 99 instances of A and only a single instance of B. Now consider a classifier that answers A in all instances. It would still yield a quite good micro F1-score:

$$F_{1,\text{micro}} = \frac{2 \cdot 99 + 2 \cdot 0}{(2 \cdot 99 + 1 + 0) + (2 \cdot 0 + 0 + 1)} = \frac{198}{200} = 0.99.$$

The macro average, on the other hand, calculates these scores for each class first. Class A reaches a score of almost one (perfect classification), while B has a score of zero, which is the worst possible F1 score:

$$F_{1,\text{macro}} = \frac{1}{2} \cdot \left(\frac{2 \cdot 99}{2 \cdot 99 + 1 + 0} + \frac{2 \cdot 0}{2 \cdot 0 + 0 + 1} \right) = \frac{1}{2} \left(\frac{198}{199} + \frac{0}{1} \right) \approx 0.4975.$$

These calculations show how the micro F1-score favors large classes over smaller ones. Class A is almost always correctly classified. But for a system that relies on a correct detection of B, this classifier would not be usable. Therefore, the micro-F1 aggregation is a good metric for a balanced dataset, but yields biased results on an unbalanced one.

2.4.10.2 AUC-ROC

For the above definition of the F1-Score, we use a fixed threshold of $t = 0.5$ for positive classifications. The choice of this parameter, however, impacts the evaluation of a given prediction model. A classifier that produces predictions with a high level of certainty (i.e. close to 0 or 1) is preferable over a predictor of the same quality which predicts values close to 0.5.

The *receiver operating characteristic* (ROC) is the curve that results from plotting the true positive rate $\text{TPR} = \frac{1}{n}\sum_{i=1}^{m} n_{\text{TP},i}$ against the false positive rate $\text{FPR} = \frac{1}{n}\sum_{i=1}^{m} n_{\text{FP},i}$ for each threshold $t \in [0,1]$. The area under this curve (AUC-ROC) yields a value in $[0,1]$. Values close to 1 indicate an almost perfect prediction with high certainty. For a random classifier, predictions and labels are independent, therefore: $P(y = 1 \wedge a_\theta(x)_i \geq t) = P(y = 1) \cdot (1 - t)$ and $P(y = 0 \wedge a_\theta(x)_i \geq t) = P(y = 0) \cdot (1 - t)$. On a balanced dataset, these values are therefore equal for all values of t. The ROC curve is then a linear function and the respective area under the curve is 0.5.

3
Related Work

As already described in Chapter 1, we will focus on two main topics in this work. Firstly, we deal with the question of how machine learning methods can be used to extend a given domain ontology. We will evaluate the deep-learning-based method we have developed using the example of the ChEBI ontology. For this specific use case, our method can be understood as a kind of chemical classification. Furthermore, we will demonstrate two ontology-based methods that can be used to help train machine learning models. Accordingly, in this chapter, we will first present existing machine learning methods for automatic ontology development and extension - in short, ontology learning. We will then discuss methods of structure-based classification of chemicals in general and then more advanced deep learning methods from the context of chemistry. Finally, we will highlight existing methods that can be used to support machine learning methods using semantic technologies.

3.1 Ontology Learning

Ontology development is time and labor-intensive and requires the involvement of domain experts from different communities. The use of artificial intelligence to support this process, also known as ontology learning [38, 39, 40, 41, 42], is therefore a natural step. However, in order to replace the involvement of experts, these systems must be able to access a suitable, domain-relevant source of information. Ontologies often represent scientific domains. In these, it is common to publish domain knowledge in books, research articles or similar. These collections of information can therefore serve as a source for the required domain knowledge. In order to extract this knowledge, however, these texts must be processed. An overview article on existing approaches to ontology learning [41] developed a comprehensive description of the most commonly used methods. As a first step, standard techniques from natural language processing are used to extract relevant terms from existing literature. The resulting collection of terms that occur within the corpus is

often too broad to be useful in ontology development. Statistical measures, such as C and NC values [43, 44] and contrastive analysis [45], are therefore used to distinguish terms that are relevant to the specific domain from those that are not.

But equally as important as domain-relevant terms are the relationships between them. These relationships can be relationships between individuals of two classes (e.g. the "part of" relationship between a car and its wheels) or logical axioms between classes such as subsumption (e.g. "Every dog is a mammal") or disjointness (e.g. "No dog is a cat"). Sanderson et al. [46] use the conditional probabilities $P(t_1|t_2)$ that a term t_1 occurs in a given document that contains term t_2. A concept referred to by a term t_1 is considered a subconcept of a concept referred to by a term t_2 if $P(t_1|t_2) > P(t_2|t_1)$ and $P(t_1|t_2)$ is larger than a predefined threshold. This method has later been extended to include multiple terms per concept [47].

Methods of subsumption prediction are often used in automated ontology extension to match extracted terms with those already present in the ontology. Althubaiti et al. [48] use a two-step approach to ontology extension. The first step aims to detect terms in a text corpus that are related to the domain of a given ontology. To do so, the model uses a text corpus as negative and embeddings of human-readable labels synonyms from the ontology as positive examples. A machine learning model is then trained on Word2Vec-embeddings [49] of these words. The second part of the system uses the "Whatizit" tool to detect ontology terms in text. A model is then trained to predict whether a given word occurs in similar contexts as classes from the ontology. These predictions are then used to predict subsumption relations to existing classes.

Cui et al. [50] use a graph-based approach to identify possibly missing subsumption relations in the SNOMED ontology. They search the graph of the subsumption relation for non-lattice pairs, i.e. pairs of classes that share at least two different superclasses that are not in a (inferrable) subclass relation. The authors argue that these structures indicate the presence of missing subclass relationships between either the two classes in the non-lattice pair or their shared superclasses. Different lexical patterns are then utilized to identify the missing subsumption relation and fix the ontology accordingly. While ChEBI contains similar structures, it does not follow as stringent naming conventions as SNOMED. A lexical analysis of labels as done in this work, is therefore not as promising. The resulting non-lattice pairs can, however, be used as a starting point for structure-based analyses.

An approach [51] that, among other kinds of axioms, also allows for the generation of existential relationships (e.g. "Every car has wheels") uses an existing

set of pre-defined predicates and concepts. These concepts are then matched with sentences in a text corpus. When two concepts occur in the same sentence and there is a known relation between these two concepts, then the sentence fragment between these classes is associated with this relationship. This dataset is then used to train several machine learning models for relationship prediction from text.

Most ontologies used in ontology learning approaches are concerned with the extension of ontologies that cover a particular domain. More general ontologies such as BFO [52] or DOLCE [53] are much broader and cover very abstract concepts with distinctions that are often complex logical or philosophical considerations. Lopes et al. [54] compare eight different deep learning methods regarding their ability to extend the DOLCE ontology based on a given term with a textual definition. While still text-based, this approach is much closer to the way we understand the process of ontology development, because it is based on the definitions from the ontology it aims to extend. These definitions reflect the consensus that ontology developers agreed upon and provides important context in which a concept is to be understood. It is however arguable, how useful this approach is in practice as it requires a textual definition not just for ontology classes, but also for concepts that it aims to extend the ontology with. In practice, there are competing, contradictory definitions for concepts external to the ontology. Finding mutually agreed-upon definitions is often the most time-consuming and labor-intensive part of the ontology development process.

Notably, all approaches discussed in this section rely on a text-based extraction of domain knowledge. Ontology development, however, goes beyond the mere extraction of existing knowledge [55]. Among the most time-consuming tasks in ontology development is the communication, comparison and alignment of different views on the same domain. This consensus is usually not present when the ontology development process starts and is, consequently, not represented in literature. A recent analysis [56] has also indicated that while ChatGPT seems to have some semblance of an upper-level ontology, it does not align with any of the existing ones (BFO, UFO, DOLCE). The authors of this study also report that the responses and the derived ontology varied significantly based on the phrasing of the queries.

A pure extraction of existing knowledge from domain literature will therefore, in the best case, lead to ambiguously defined concepts that hinder the consistent annotation of data or, in the worst case, lead to an ontology that is logically inconsistent. The approach we present in this thesis tackles the problem from a different angle. We aim to extract the existing consensus from an existing ontology and train a system to adhere to that consensus.

3.2 Chemical Ontologies and Structure-Based Classification of Chemicals

Domain ontologies are - as the name implies - developed to describe a certain domain. For our study, we will focus on an ontology from the domain of chemistry, namely the ChEBI reference ontology, discussed in Section 2.2. ChEBI has been widely adopted and can be considered the "gold standard" chemical ontology in the public domain. It is used for informatics applications such as bioinformatics and systems biology analyses of metabolism, biological data integration, natural language processing, and as a chemistry component for semantic web applications (e.g. [57, 58, 59, 60]).

Although chemical classification is typically based on structural features, full computable structural definitions for classes – which would support the use case of ontology extension through automated reasoning – are seldom formally captured in chemical ontologies, as the underlying logical formalisms are not able to encompass the full range of relevant structural features [61, 62, 26]. Thus, chemical ontologies typically only reflect a partial axiomatization for the chemical domain. Various extensions to the OWL language have been explored in order to allow a fuller set of structural features to be exposed to the computable classification hierarchy, including description graphs [63], logic programs [64], and non-monotonic existential rules [62].

Nevertheless, these formalisms have not seen widespread adoption, in part because they are not supported by the wide range of tools that are available for OWL, and in part, because performance remains a challenge. Thus, they have not been able to scale to the use case of extending real-world ontologies such as ChEBI. Moreover, chemists themselves often only have implicit knowledge (e.g. about which functional groups or structural patterns are most relevant for a classification) that is hard to make explicit.

The SMILES language has an extension, SMARTS, which allows ambiguities and patterns in, and logical combinations of, chemical structural elements, to be represented in a way that allows for the definition of classes of chemicals. This language has been used to define chemical classes in chemical ontologies such as OntoChem's SODIAC [65] and more recently the ClassyFire application [25]. At the time of writing, ClassyFire is the state-of-the-art tool for structure-based chemical ontology classification, in terms of size (9,000 definition rules, and an associated ontology of 4,825 classes) and adoption, and is used in the automated extension of the ChEBI ontology. However, as we discussed in Section 2.3.3, ClassyFire's scope is limited and the tool itself is not open. Thus, the tool operates as an algorithmic "black box", in that the rules it uses are not semantically accessible, the associated chemical ontology

still has to be maintained manually, and updating the integrated knowledge system can only be accomplished by updating the custom software suite.

The expressivity of SMARTS is, also, still limited. In research [66] tangential to this thesis, we drafted an approach that extends ChEBI with axioms in first-order logic and used automated reasoning for classification. While first-order logic is more expressive than SMARTS, it has been shown that it is still not strong enough to capture important molecular structures, such as connectedness [67]. More expressive logics such as monadic-second order logic are needed [26] but reasoning for these expressive logics is inefficient.

However, the limitations of current formal approaches to describe molecules pose a significant hurdle to their usability for ontology extension tasks. The detection of different kinds of features is relevant for the purpose of extending ChEBI because the ontological distinctions in ChEBI's taxonomy can be based on molecular structure, function or the chemical origin of a chemical entity. An automated tool that aims to extend ChEBI therefore needs to detect these orthogonal concepts for which formal specifications are often incomplete. Machine learning methods have been successfully employed for many tasks, for which a formal specification is difficult or infeasible. Such methods are therefore important for an automated extension method of ChEBI.

3.3 Machine Learning and Deep Learning Approaches

Applications of artificial intelligence in the domain of chemistry have been a heavily researched topic in recent years with a major focus on the prediction of specific application-centered targets. Maziarka et al. [68] use an adjacency-based interpretation of the attention mechanism in a transformer-based system for classification and regression in order to predict properties like solubility (FreeSolv [69], ESOL [70]), the ability to penetrate the blood-brain barrier (BBBP [71]), activity towards estrogens, and metabolic stability (MetStab [72]). Another transformer-based approach [73] uses an auto-encoder to build a latent space that is then used to train Gaussian process and support vector regression models in order to predict the boiling point of organic fluids. Deep learning has been used in chemistry in various ways, e.g. for protein structure prediction, drug design, property prediction and even synthesis planning [74, 75]. The DeepChem library [76] is an open-source machine learning framework that has gained widespread adoption in the life science community as it offers a variety of tools and algorithms to facilitate chemical property analysis. MoleculeNet [77], which serves as a benchmark and curated dataset collection for molecular machine learning, is also released as a part of the DeepChem library.

Deep learning has also been used for the classification of molecules in chemical ontologies (which in turn can be used for ontology extension). In [78], a back-propagating artificial neural network is applied to classify natural products, that is, secondary metabolites largely of plant origin. Named NPClassifier, it is trained on a dataset of around 73,000 natural products sourced from public databases including PubChem, ChEBI, and the Universal Natural Products Database. The hierarchy into which these molecules were organized consisted of three hierarchical levels: 7 *Pathways*, 70 *Superclasses*, and 653 *Classes*. Rather than training a single model for the full prediction task, they used three single-task models – one model for each of the classification hierarchical levels. They report promising performance in a direct comparison to ClassyFire for a selection of classes. However, the restriction to only natural products (a subset of organic molecules) and to only three hierarchical levels addresses an artificially simpler task than the general problem of classification in chemical ontologies, where classes can be arranged in a hierarchy of arbitrary depth and reflect a wider chemical diversity.

In [79], machine learning was used to predict class membership directly from mass spectrometry features in an untargeted metabolomics study. This is an important use case, as in untargeted metabolomics there are often many features that relate to 'unknown' molecular entities and thus are not mapped to defined molecular entities about which metabolic information is known, however, they may nevertheless share detectable chemical classes. In this effort, the chemical fingerprint was used as an intermediary structural representation for learning purposes: support vector machines were used to predict chemical fingerprints from mass spectrometry features, and a deep neural network was then used to predict class membership from the resulting fingerprints. However, their system predicts class membership only for a subset of the classes belonging to the chemical ontology underlying ClassyFire and moreover does not attempt to extend the ontology itself.

A particular focus of deep learning approaches lies in the prediction of properties that are hard or expensive to test, like toxicity [80, 81], have been in focus of these approaches. The Tox21 challenge and the datasets that have been derived from it have been a benchmark for the evaluation of many machine learning systems. The "Toxicology in the 21st Century" (Tox21) program hosted a competition [82] in 2014 on the challenging task of toxicity prediction, in which 40 different approaches participated. The winner of this challenge [83] is based on a deep feed-forward network with up to four layers in an ensemble with support vector machines, random forests and a variant of a linear regression classifier. Their system pre-calculates a significant number of features, including the absence or presence of "2500 predefined toxicophore features, i.e., patterns of substructures previously reported as toxicophores in

the literature" [83]. They also note that extensive cleaning was necessary before using the original training set. The inputs of the system were often "mixtures of substances" instead of individual molecules. These structures may be salts or other structures that were loosely connected via ion bonds, but not covalent bonds. The authors split these into their connected components and considered them as individual substances, whose predictions were then merged via a mediation process. Further data augmentation techniques were applied to counteract the limited size of the Tox21 dataset. This need for data augmentation has been a common theme throughout the successful participants of the Tox21 data challenge. Since then, a new, cleaned version of the Tox21 dataset has been published as part of the MoleculeNet [77] data collection. We will use both Tox21 datasets in a later evaluation.

One approach that used ontologies for toxicity prediction [84] for proteins, is based on the Gene Ontology (GO). Here, the ontology is used to generate feature vectors as input for a deep neural network in order to predict a more general notion of toxicity than the one used in this work. Proteins are extracted from a GO-annotated database with their respective annotations. These annotations are then used to construct a binary vector for each protein that represents the presence or absence of a particular annotation.

The embedding of definitions can be used to understand concepts [54]. There are also more general approaches to embedding natural language in high-dimensional spaces. Word2Vec uses either a continuous-bag-of-words or skip-gram methods to achieve such embeddings. It was shown that important semantic relations are preserved in the spatial relations of the embeddings, even if they were not explicitly present in the training data. The question of whether such embeddings can also be considered for ontology terms was pursued further by Onto2Vec [85]. Axioms are represented as syntactic sentences and the ontology is a text consisting of several sentences. These embeddings of classes are then used to embed a given new instance as the sum of all its associated class embeddings. Based on a similarity measure on instances, a system could then be developed that predicts protein-protein interactions with a quality that is only slightly inferior to that of humans. In its extension OPA2vec [86], this approach is extended to also include text-based annotations. While based on ontological knowledge, this approach cannot be easily applied to ontology extension. Onto2Vec and OPA2Vec require existing ontological annotations and axioms to generate a prediction for a given entity. In this work we seek a system that can generate these axioms. However, there is a potential synergy between the two systems in that a combination of both can be used to predict the properties of new proteins, which is worthy of further investigation in future work.

3.4 Training with Semantic Support

Researchers are also increasingly interested in semantic validations in the wake of growing concerns about the logical inconsistencies in responses from systems like ChatGPT but the principles of combining machine learning and semantic methods are not new. “Knowledge-Based Artificial Neural Networks” (KBANN, [87]) attempted to directly represent formulae in propositional logic within the network structure. During training, the system is able to adapt these structures to better fit the training data. This allows the priming of a learning system with prior knowledge.

The training process of neural networks is usually based on a form of gradient descent. Consequently, in order to allow answers as truth values $\{0,1\}$, one must allow arbitrary predictions from $[0,1]$ in order to retain differentability. This naturally leads to an interpretation of these values as values from a many-valued logic such as fuzzy logic or probabilistic logic. Indeed, there have been many approaches that aim to combine fuzzy systems and neural networks [88, 89]. These systems are particularly useful when training data is limited. In a recent work [90, 91], we applied an ontology-based neuro-fuzzy controller. The approach in this section was inspired by this work, in which we also applied a semantic penalty system to ensure logically sound rules.

DeepProbLog [92] is based on a probabilistic interpretation of these prediction values. This approach is based on the probabilistic logic programming framework ProbLog [93]. ProbLog allows the expression of Prolog-like inference rules with additional uncertainty annotations, e.g. *0.3::P(X) :- Q(X), R(X).* The formulation of these rules, however, requires extensive expert knowledge or data in order to derive the appropriate annotations. DeepProbLog extends this framework by allowing these uncertainty annotations to be derived from a neural network. DeepProbLog relies on a model counting approach in order to derive the probability for a given atom.

Logic Tensor Networks [94] follow a similar approach to DeepProbLog. Here too, predicate interpretations are represented by neural network operations. However, Logic Tensor Networks do not use probabilistic semantics here, but fuzzy semantics. A grounding function $\mathcal{G}$ interprets constants as vectors in an n-dimensional space and predicates as tensors. In their original version, a sentence $P(a,b)$ is then assigned a fuzzy truth value based on the tensor product with a concatenation of the grounded terms $\mathcal{G}(a)$ and $\mathcal{G}(b)$. In a more recent publication [95], this approach is extended to allow for arbitrary representations of predicates as operators that can be manually defined or be represented by arbitrary trainable neural networks.

Neural networks are, in particular during training, prone to make mistakes that may result in logically inconsistent predictions. An image recognition system may, for example, classify the same picture as a cat and a dog. In combination with logic approaches, these mistakes may cause severe side effects or other systems that expect consistent input [96]. In most classical logics, once an inconsistency has been derived, any statement can be inferred. This strong effect of inconsistencies is not desirable in applications that must allow for some level of inconsistency - in particular, if human input is used. If a person makes an inconsistent statement in their tax form, a possible neuro-symbolic tax system should not be able to infer that Elvis is the king of Sweden or other arbitrary facts from that - the inconsistency should be kept local. Logical Neural Networks [97] allow for some local inconsistencies in their reasoning process. This kind of network is designed to directly represent the structure of a logical theory with upper and lower bounds instead of truth values. During inference, these systems also use a semantic penalty term that trains the system to avoid logical inconsistencies.

For the problem of protein function prediction exist several approaches that use the Gene Ontology [4]. The DeepGO approach [98] uses a combination of convolutional networks and knowledge graph embeddings that are combined into a hierarchical classifier to predict protein functions from the Swiss-Prot knowledge base [99]. This approach is an excellent example of how an ontology can be incorporated into the learning process. Here, the rich knowledge base of the ontology is used to additionally ensure the consistency of the target labels. This is done by using a special ontology-based output layer. DeepGO was later extended in scope and architecture to the DeepGOPlus [100] model. Herein, the convolution network has been changed to a more compact version with multiple filters and the model uses a flat output representation instead of the hierarchical classification layer. The semantic structure of the Gene Ontology then returns to the focus in the follow-up work DeepGOZero [101]. Despite the similar name, this approach is not based on convolutional networks but instead represents GO classes that represent protein functions as n-dimensional balls and relations as vectors using ELEmbeddings [102] and a lightweight two-layer multi-layer perceptron as input embeddings that represent proteins as n-dimensional points. Whether a protein has a certain function is then determined by its containment in the ball that represents that class after being shifted along the vector that represents the *has function* relation. This allows predictions for classes that have not been seen by the model before and can therefore be used for zero-shot learning.

The starting point for this work is as follows: Effective ontology development requires suitable tool support. While the ChEBI ontology already uses a tool for development in the form of ClassyFire, this is not customized to ChEBI and

not actively developed. Formal methods are theoretically suitable for representing at least the structure-based necessary classification criteria, but complex, logical formalisms are required that are not yet widely established in the domain and for which there is only limited tool support. Another problem with purely formal approaches is the need for manual extension.

Therefore, in this work, we want to take a closer look at the methods of automatic ontology extension. Approaches based on machine learning already exist. However, these are based on domain literature, which often shows ambiguities and inconsistencies. An analysis of large language models also indicates that these inaccuracies are transferred to the models [56]. Instead, we look for methods that utilize the existing structure of the ontology to find a predictive model for possible extensions. To our knowledge, no such system exists yet.

Furthermore, ontologies are an excellent source for learning domain knowledge. It is easy to see that using this knowledge for learning tasks is beneficial. In particular, we have already discussed methods from the field of protein functions that successfully apply ontology-based knowledge. The question therefore arises as to whether a model that has understood the consensus criteria of an ontology can also use this knowledge for further tasks and whether the ontology can be used in other, more direct ways.

4

Ontology Extension

In Section 2.2, we discussed the current state of the ChEBI ontology. The difference in speed between the pace with which ChEBI and PubChem are extended highlights that ontology development needs to be supported by fully automated or semi-automated methods. One of the significant challenges for automated ontology extension approaches is the need for precise domain knowledge in order to determine the necessary distinctions that need to be made to add a new concept to an existing ontology. Most approaches to ontology extension rely on domain literature to deliver this important knowledge. One of the core purposes of an ontology is however to properly define the key distinctions for its target domain. Different literature sources from a domain that has not yet adopted a shared conceptualization are therefore prone to contain inconsistent nomenclatures and definitions that even domain experts may not be aware of [55]. Using these sources as a basis for automated ontology extension will transfer these inconsistencies to the ontology.

To address this, we propose a system that requires no external resources and is built based only on information that is contained within the ontology. This requires that the ontology adheres to a set of specific constraints. Recalling the ontology definition from [16], an ontology consists of three parts: A vocabulary, its annotations and a logical theory. In order to appropriately extend an existing ontology, the system needs to learn the intended meaning of classes that already exist. We discussed earlier that a vocabulary that is tailored to one of these subcommunities will not match the usage by others. A vocabulary on its own is therefore not an appropriate source to understand the intended meaning of an ontology. Logical axioms are better suited to capture the semantics of concepts. The logical theories underpinning ontologies are often too weakly axiomatized to fully capture the intended meaning of all classes. "A dog is a mammal" does not provide any insights into what dogs or mammals are. Whilst there are efforts to build ontologies based on more expressive logics such as the first-order axiomatization for BFO, most ontologies are comprised of little more than simple subsumption and disjointness

axioms. Complex formalizations also often limit the readability of an ontology due to the inherent complexity of expressive formal logic languages. The best link between ontology concepts and their intended meaning lies in annotations. For most ontologies, these consist of natural language definitions for all classes. However, this requires a system that is able to understand definitions. In order to derive the intended meaning of a class from its natural language definitions, it needs to understand the intended meaning of all words that are used therein. Ontologies often use so-called Aristotelian definitions, in which a concept is defined in relation to the closest concept by which it is subsumed. "A dog is a mammal that has a tail, teeth and that barks" defines the concept of "dog" in relation to its superclass "mammal" and also other concepts such as "tail", "teeth" and "bark". A system that is built to derive the meaning of "dog" purely from its textual definition would require it to already know the meaning of "mammal". This could either be derived from literature with all the disadvantages that we discussed earlier. Alternatively, the meaning of missing terms could be derived from the definitions annotated within the ontology. This results in a recursive cycle until there are either classes that have no definitions or there are dependencies between definitions that are circular.

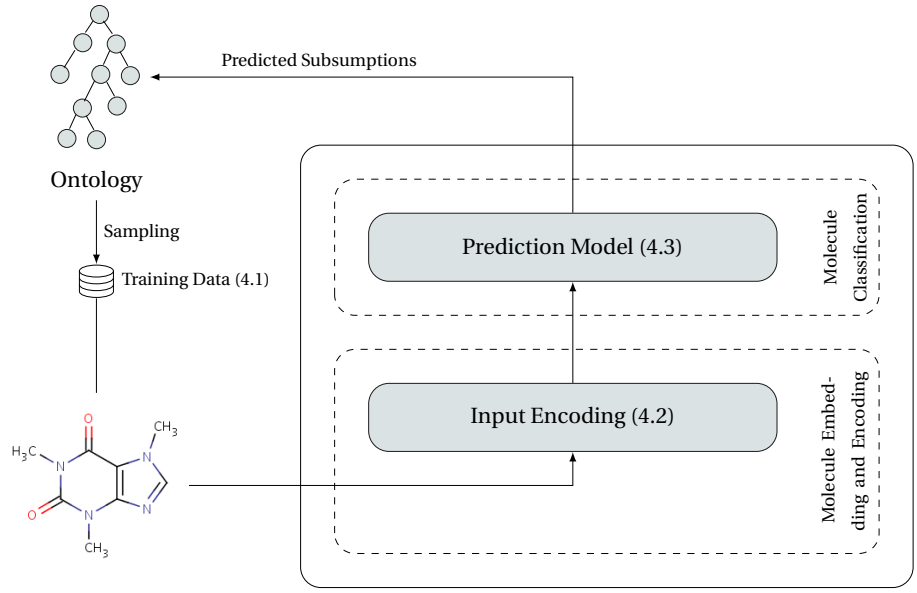


Figure 4.1: Schematic depiction system proposed in this work. The modules are annotated with those sections that describe them.

In this work, we aim to overcome these problems by utilizing additional *structural annotations*. We define structural annotations as any additional annotation that is machine-readable and characterizes instances of this particular

class. These machine-readable annotations give more detailed insights into the intended meaning of a class. Examples of these structural annotations are the SMILES annotations we discussed in Section 2.3.1. If a ChEBI class is annotated with a SMILES string, all instances of this class are required to adhere to this structure and all SMILES annotations of subclasses are refinements of this SMILES string.

In this chapter, we present a system that uses structural annotations in order to predict new subsumption relations for a given ontology. Figure 4.1 depicts a schematic representation of the proposed model. In the following chapter, we will present the different modules of this architecture in the sections denoted in the figure.

This thesis is the result of joint research work with my collaborators. However, the resulting methods and results are only really meaningful in their entirety and to limit the analysis to those results would obscure important context. Nevertheless, in order to make my contribution to the work transparent, I will insert a box labeled "My contributions" at the beginning of each relevant section. In these, I will explain what work the reported results are based on and what part of the work I have done.

4.1 Data preparation

My contributions

The following section summarizes the datasets used for our ontology extension task [103, 104, 105, 106]. My contribution was the construction of all multi-label and pre-training datasets, with the exception of Pubtox and the $\text{ChEBI}^{709}_{v148}$ datasets. The inclusion of these datasets into the ChEB-AI framework, however, was also done by me.

All datasets in this chapter are derived from the ChEBI ontology, which is publicly available. In particular, we use version 200 of the ChEBI ontology. In order to prepare the ChEBI ontology for the learning tasks, we downloaded and parsed the OBO format export of ChEBI's ontology. We use only the hierarchical (*is a*) relationships from the ontology for this work.

As we discussed in Section 2.2, ChEBI's classes define sets of individuals, even if their respective SMILES annotation is fully specified. There are however classes that allow for heterogenous structures within their instances and those that only cover homogenous structures. For the purpose of the data aggregation, therefore, introduce the distinction between *partially specified classes*, which we define as those classes within ChEBI that subsume multiple struc-

turally annotated members, *fully specified classes*, which we define as those classes in ChEBI that are associated with a fully-defined molecular structure, indicated by the presence of an associated SMILES annotation that does not contain wild-cards.

The number of classes within the full ChEBI ontology, their unbalanced sizes, and the problem of multiple inheritance at every level make it challenging to train some classifiers on the whole ontology, in particular simple classifiers that predict just one class for a given chemical structure as input. Other, more modern approaches do however tend to benefit from larger datasets. We thus implemented several different data selection schemes to compare different approaches.

4.1.1 A Balanced Approach

Traditional classification methods assume a more or less homogenous distribution of members for each class. In order to support approaches that benefit from balanced data, we developed a dedicated selection strategy that does not use the full ChEBI ontology but rather chooses classes, and sub-samples randomly from their members, such that the result is *balanced* (i.e. all class having a similar number of members). We restrict the sampling of classes to those that are classified beneath the 'molecular entity' root of ChEBI, as this is where the bulk of the leaf members with defined molecular structures are found.

Alongside the need to prepare a balanced dataset in terms of the number of members per class, it is also important that the members with structures are selected so that individual members are not duplicated across multiple classes, in order to enable the clean separation of the dataset into training, test and validation sets. In practice however, the ontology contains a large percentage of overlapping members between classes, since the ontology classes higher up in the hierarchy describe general chemical features that in many cases can be compositionally combined in classes lower down in the hierarchy [61], as illustrated in Figure 4.2. To mitigate this challenge, the selection process only sampled each leaf member structure once, assigning it as a member of the training data for just one class, even though in the actual underlying ontology that molecule in fact belongs to multiple classes. This is an artificial restriction for the purpose of the learning task: we sub-sample the leaf members with molecular structures for each class such that no leaf member with structure is selected for more than one class. Therefore, each molecule in this dataset is assigned exactly one class and, thus, the learning tasks on this dataset are multi-class learning problems.

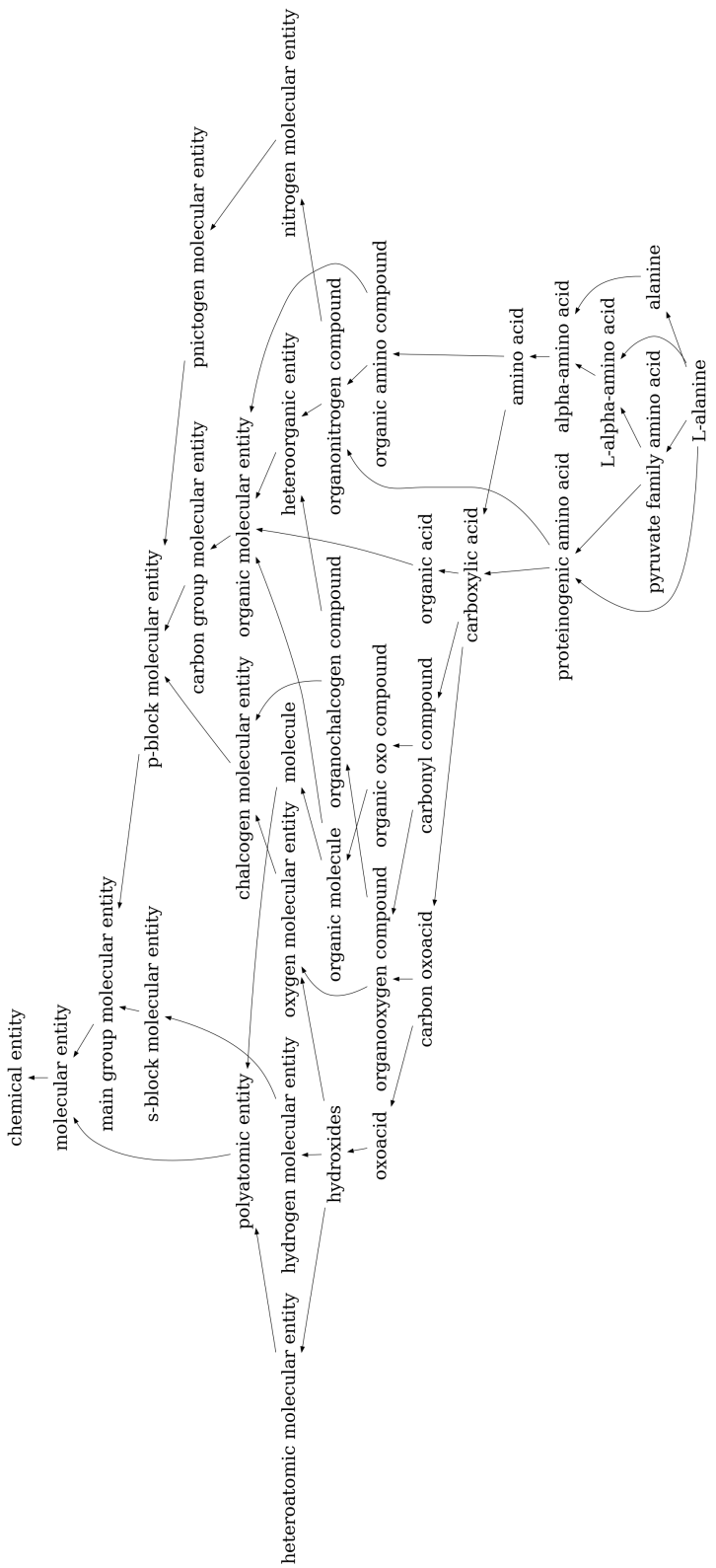


Figure 4.2: Class hierarchy for the molecule *L-alanine* (CHEBI:16977) with many branching and partially overlapping ancestor. Arrows indicate that the class at the source is a subclass of the class at the target.

Sub-sampling members for classes such that no classes have any shared members is likely to introduce a bias. We designed the selection in a way that will have the least impact by minimizing the discrepancy from the actual ontology structure. The classes were first sorted from the smallest to the largest (in terms of the number of leaf members with structural annotations) to prioritize classes with fewer members over classes with more members. This reduces the amount of sub-sampling required. Following this strategy, we iteratively selected sets of N classes with M randomly sampled member structures, where N and M were specified dynamically, to be able to evaluate performance across a range of different problem sizes. A dataset containing N classes of which each has M members will be denoted as $N \times M$. No additional chemical prioritization strategy (e.g. to ensure chemical diversity in selected members) was used in the selection.

4.1.2 Lifting Data Limitations

The formulation of the classification of chemicals as a multi-class problem does not appropriately represent the way ontologies are structured. Due to the subsumption relation between classes, a class that is not a direct subclass of *chemical entity* (CHEBI:24431) has at least inferable superclasses. Therefore, we aimed to lift this restriction for all the following datasets and turn the problem into a multi-label classification task. Herein, a label is positive, if and only if it represents a superclass of a given molecule. We derived multiple datasets as depicted in Figure 4.3. All multi-label datasets follow the naming scheme $\text{ChEBI}^{[\text{number of labels}]}_{[\text{ChEBIversion used}]}$. The first is a multi-label-version of the dataset 500×100, which retains the same 500 classes but assigns instances in the dataset to all classes of which they are a member according to ChEBI. In the following, we will also call this the $\text{ChEBI}^{500*}_{v200}$ dataset. The star indicates that due to the previous selection process, contrary to all of the following datasets, this dataset does not contain all possible members that are in ChEBI.

The balanced data extraction has specifically been designed for those methods that are vulnerable to differences in representation for different classes. Yet, this approach also limits the amount of data that is available during training. In order to analyze the impact of this limitation, we created a second dataset including the same classes as in the previous dataset, but for each class including all possible molecules that belong to this class in ChEBI, i.e. without sub-sampling only 100 members for the selected classes. The resulting $\text{ChEBI}^{500}_{v200}$ dataset shows greater levels of imbalance and overlap but features almost twice as many training instances. By using both datasets, we are able to contrast the detrimental effect of imbalances vs. the positive effect of

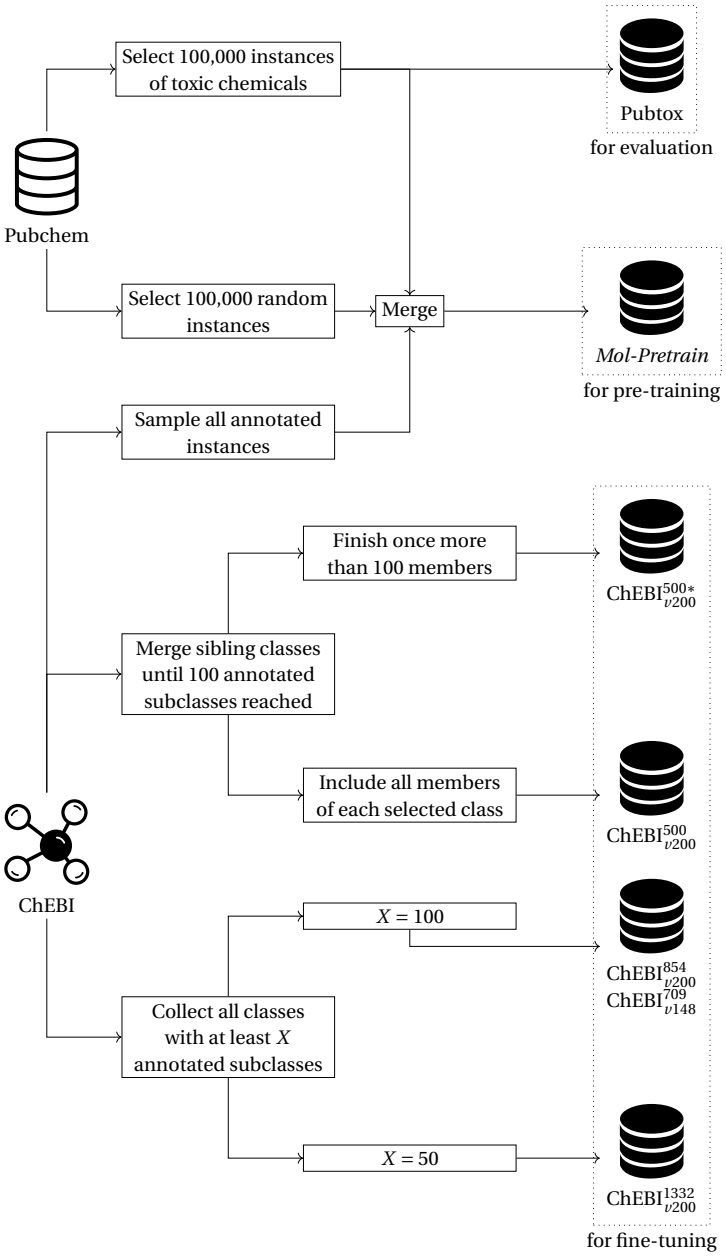


Figure 4.3: Processes used to derive the multi-label datasets from ChEBI and Pubchem. The dotted boxes group datasets w.r.t their usage in the ontology extension task. ChEBI$^{854}_{v200}$ and ChEBI$^{709}_{v148}$ are derived from the same process, but from different versions of ChEBI.

increasing the dataset size for learning. Figure 4.4 depicts the differences in member distribution and the increased imbalance between these datasets.

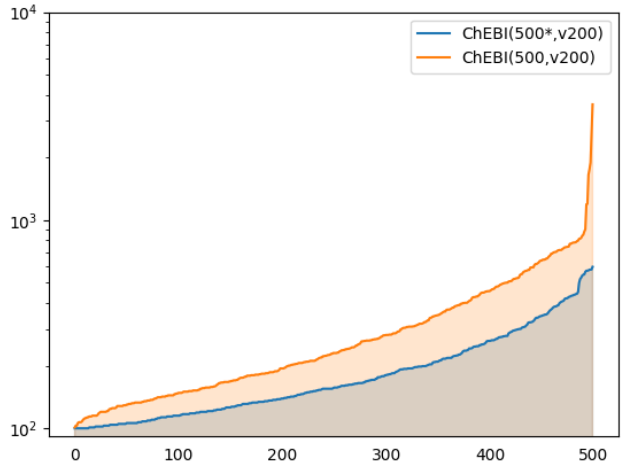


Figure 4.4: Distribution of class sizes on the $\text{ChEBI}^{500*}_{v200}$ and $\text{ChEBI}^{500}_{v200}$ dataset.

While the focus of the $\text{ChEBI}^{500}_{v200}$ dataset was to analyze the impact of a larger, more imbalanced dataset on the same classes as the $\text{ChEBI}^{500*}_{v200}$ dataset, we also wanted to investigate the performance on a larger fragment of ChEBI. The limitation on those classes represented in $\text{ChEBI}^{500*}_{v200}$ and $\text{ChEBI}^{500}_{v200}$ puts a larger focus on those classes that are further down in the hierarchy. The $\text{ChEBI}^{854}_{v200}$ dataset contains all classes in ChEBI that have at least 100 annotated subclasses the resulting dataset raised the number of classes to 854. We also created a version of this dataset with a lowered threshold of 50 resulting in the $\text{ChEBI}^{1332}_{v200}$ dataset with 1,332 classes.

The $\text{ChEBI}^{709}_{v148}$ dataset has been constructed in the same way as the $\text{ChEBI}^{854}_{v200}$ dataset but the generation was based on ChEBI version 148, instead of version 200. This version contained only 709 classes with 100 members. It was ensured that the test sets contain the same instances, so far possible, by discarding all SMILES-annotated classes that are present in the $\text{ChEBI}^{854}_{v200}$ test set before creating a train-validation split for $\text{ChEBI}^{709}_{v148}$. Therefore, this test set can be used both for models trained on $\text{ChEBI}^{854}_{v200}$ and for models trained on $\text{ChEBI}^{709}_{v148}$. For the evaluation of models trained on the $\text{ChEBI}^{709}_{v148}$ dataset, only the labels of the test set have been changed to the 709 labels present in $\text{ChEBI}^{709}_{v148}$. Notably, only 701 classes can be found in both $\text{ChEBI}^{709}_{v148}$ and $\text{ChEBI}^{854}_{v200}$. This discrepancy stems from the ongoing development that removed some classes from the ontology.

4.1.3 Going Beyond ChEBI

The great advantage of ChEBI is its rich semantic foundation, which, however, can only be guaranteed with a considerable amount of work. However, this structure is not necessary for certain learning tasks that are not based on automatic ontology-based classification. Therefore, we also generate a data set that is composed of PubChem's much larger, unstructured pool of chemical compounds. Whilst PubChem has a much larger variety of different chemicals, molecules are not necessarily associated with their chemical classes in ChEBI.

We want to test our method of ontology extension with a dataset of chemicals not yet present in ChEBI that is based on a plausible real-world use case. We selected all chemicals from PubChem that have a hazard class annotation (n=152,205, as of October 2021). These can be assumed to be biologically relevant and well within the target scope for ChEBI, but as a group are not yet well represented in ChEBI. We then excluded any that were already present in ChEBI, which reduced the overall number to 140,913. The resulting Pubtox dataset serves as an exemplar of a set of molecules that would typically form the use case for ontology extension.

This unlabelled dataset can also be used in learning tasks that do not require present labels, such as pre-training. To further extend the variety, we also randomly sampled 100,000 molecules from the broader PubChem database and merged the result with the previously discussed "hazardous" dataset and all molecules from the ChEBI ontology. The resulting *Mol-Pretrain* dataset has been used to pre-train all Electra-based models.

4.2 Input Encodings

My contributions

The following section summarizes methods used to process the annotated SMILES strings into data formats that are useable by different methods based on collaborative work [103, 104, 105, 106]. My contribution was the conceptualization and implementation of the *Char* and *Chem* tokenizers.

The data sets presented above are all based on SMILES representations. However, these cannot be processed directly by all of the approaches that we want to discuss in this work. Therefore, in this section, we would like to present various methods with which these SMILES representations can be preprocessed.

4.2.1 Fingerprints

In Section 2.3.2 we introduced the notion of chemical fingerprints for molecules. For each structure, we calculated such a fingerprint using the RDKit [24] software library's RDKFingerprint, represented as a bit string with a size of 1024 bits. For the maximal neighborhood depth, we used the default value of $k = 7$.

4.2.2 Tokenization

For learning systems that require fixed-length inputs, as do many of the classifiers that we tested, fingerprints are a feature-rich input encoding that has regularly been used. However, encoding structural features via fingerprints may lose crucial information concerning the actual arrangement of these features. Certain kinds of artificial neural networks, such as transformers, are able to process variable-length inputs. SMILES can be regarded as a language with atoms and their bonds as the alphabet. In connection with the learning objectives for our classification problem, language models have shown to have the potential to be applicable for chemical classification [107]. Thus, we also explored using the full SMILES representation. Similar encoding approaches have been successfully employed in analyses of natural language [108].

We have also used this simplest tokenization method for the SMILES strings in this work. But even in natural language, letters are not very helpful on their own. Certain word fragments often have similar functions. In English, for example, the prefix "un-" negates adjectives before which it appears. Accordingly, linguistically meaningful or frequently occurring fragments are often combined during tokenization. Similarly, do Hydrogen (H) and Helium (He) or Phosphorus (P) and Lead (Pb) have very different properties despite their overlapping syntactic elements. Therefore, it may be beneficial to remove the burden of learning these differences from the dataset by using an encoding that is based on fragments of the SMILES string.

C	C	1	(	B	r	)	C	(	S	c	2	c	c	c	c	c	2	[	N	+	]	⋯

(a) Tokenisation without atom grouping

CC	1	(	Br	)	C	(	Sc	2	ccccc	2	[	N	+]([	O	-])=	O	⋯

(b) Tokenisation with atom grouping (BPE)

C	C	1	(	Br	)	C	(	S	c	2	c	c	c	c	c	2	[N+]	(	[O-]	O	⋯

(c) Tokenisation with atom grouping (Chem)

Figure 4.5: Comparison of tokenizations for a fragment of the SMILES strings of *BNPS-skatole* (CHEBI:85968)

Sub-word tokenization strategies, such as Byte Pair Encoding (BPE), break a text sequence into sub-words that can then be recombined to generate embeddings for previously unseen words. The BPE algorithm begins by counting the number of times each character pair appears in the dataset. After each iteration, the most frequently occurring pairings are merged and added to the vocabulary. For the next iteration, the two characters in each pairing are combined and considered as a single unit. The performed merge operations are stored and used when the tokenization algorithm is confronted with an unknown token. So, by considering the possible merges for the character pairs in an unknown token, this unknown token can be broken into smaller known tokens. As previously discussed, this approach makes sense when processing natural language, as it is compositional in nature [109] and similar word components tend to have similar meanings. However, this can only be transferred to the context of chemistry to a limited extent. Since the BPE tokenizer is based on subword frequency, it does not respect the intended semantics of the chemical abbreviations. The token vocabulary frequently contained SMILES-specific reference characters that capture branches, cycles or molecule charges resulting in chemically infeasible tokens. Figure 4.5 exemplifies the tokenization of a SMILES string the tokenizers used in this work. It can be seen that the BPE tokenizer merges a charge indicator (`"-"`) and an adjacent double bound into a single token (`"-])="`). The BPE technique does also not allow deeper insights into the influence that some singular atoms had on the model's predictions. Thus, given a token *OCB* it would not be possible to ascertain whether the model based its prediction on Oxygen, Carbon, Barium or a combination of those. It is also possible that the probability-based merging of tokens conflates tokens that should not necessarily be merged into one. Figure 4.5b shows how the contained sulfur atom (`S`) has been merged with the adjacent carbon atom (`c`) into a token that one would expect to represent scandium (`Sc`). This limits the interpretability of token-based analysis approaches.

Therefore, we developed a third tokenizer that parses the SMILES string according to its concrete syntax and uses the resulting syntactic categories as tokens. This means that tokens consist of single atoms including their respective charges and isotopes (e.g. `"[Br]"`, `"B"`, `"[Fe2+]"`), bonds (e.g. `"-"`, `"="`) and ring and branch references. However, the current encoding has no special handling for different charges of the same element, i.e. `"[Fe2+]"` and `"[Fe3+]"` are embedded independently. Figure 4.5c shows an example using the resulting *Chem*-tokenizer.

4.3 Prediction Model

My contributions

The following section summarizes the different models used for the ontology extension task based on collaborative work [103, 104, 105, 106]. My contribution was the implementation and training of the *LSTM* and *Electra*-models as well as the implementation of the ChEB-AI framework.

In the previous sections, we have presented the individual building blocks that are necessary for understanding this work. Now we want to show how these are put together to form an overall system. To this end, we will further refine the system described in Figure 4.1 and explain how we combine the encodings described in Section 4.2 with the architectures introduced in Section 2.4 to ultimately predict subclass relations for an ontology.

4.3.1 Traditional Approaches

Out of the classification algorithms that we presented in Sections 2.4, seven require input lengths of a fixed size, namely Logistic Regression, k-Nearest Neighbor, Decision Trees, Ransom Forests, Naive Bayes, Linear Discriminant Analysis and Support Vector Machines. This poses a challenge for applications in chemistry, as molecules occur in a variety of different sizes. Accordingly, the SMILES annotations on the ChEBI classes also have different lengths and cannot be used as input for these models without further processing. So we use the topological fingerprints presented in Section 2.3.2 to convert the SMILES strings into fixed-size inputs. All these methods were used as implemented in the scikit-learn library [110] and their performance has been evaluated on the multi-class problem that is represented by the balanced $N \times M$-datasets described in Section 4.1.1.

Logistic Regression Logistic regression, at its core a binary classification algorithm. In order to apply it to a multi-class problem, we use the *One-VS-Rest*-methodology. This method also allows for a separate weighting of classes to address possible imbalances. This balancing is however not necessary for the dataset used in this category, because all classes are by design balanced. We used the *liblinear* solver as an optimizer and L2-Regularization to counteract overfitting.

Support Vector Machine In this work, we used Support Vector Machines with three different kernels. The first, linear kernel $K(x_i, x) = \langle x_i, x \rangle$ aims to separate the problem space using a linear plane – similar to linear regression – but adds additional penalties for points that are too close to the line of separation. This results in a more robust classification. The second, sigmoid kernel $K(x_i, x) = \tanh(\langle x_i, x \rangle)$ reshapes the problem space with respect to a sigmoid function (e.g. tanh). This makes points on the high end of the function more easily separable from the ones on the low side. And finally, the radial basis kernel $K(x_i, x_j) = e^{-\|x_i - x\|^2}$ reshapes the problem space according to a Gaussian distribution around similar points in the feature set. All three Support Vector Machines have been trained with L2-Regularization.

***k*-Nearest Neighbor** We used the standard Euclidean distance to determine neighborhoods of size five. Notably, because the feature space for the topological fingerprints used is binary, this metric is equal to the square root of the hamming distance of the topological fingerprints. For a given data point, the classification has been derived by an unweighted majority vote, i.e. that class that was represented in most of the members of the point's 5-neighborhood, was picked as the predicted class. In case of a tie, the class with the smaller index is assigned.

Decision Trees/Random Forests We used the Gini impurity as a decision criterion to pick the best splitting points among all available features. As with logistic regression, we did not employ any weighting scheme for different classes. These trees were used as single predictors or in a random forest, i.e. as an ensemble of 100 decision trees. Individual trees in this ensemble were only trained on a subset which size equaled the square root of the size of the total training set.

Naive Bayes/Linear discriminant analysis For Naive Bayes and Linear Discriminant Analysis, we used a Gaussian distribution. Additionally, the LDA is combined with a Singular Value Decomposition to reduce the number of dimensions to the number of available classes minus one.

4.3.2 Sequence-based models

The sequence-based models in this work used the SMILES annotation from ChEBI as input. We used the tokenization approaches described in Section 4.2. Figure 4.6 illustrates the way we apply the LSTM model to a given tokenization. Each token is embedded into a real vector. The LSTM consumes each of these embeddings in sequence and adapts its internal activation and memory state.

The output of the last cell is then passed through a dense layer with a trailing dropout to prevent overfitting. The results are passed to a dense layer with a sigmoid activation function. The result is a vector of values between 0 and 1, representing predictions of this model.

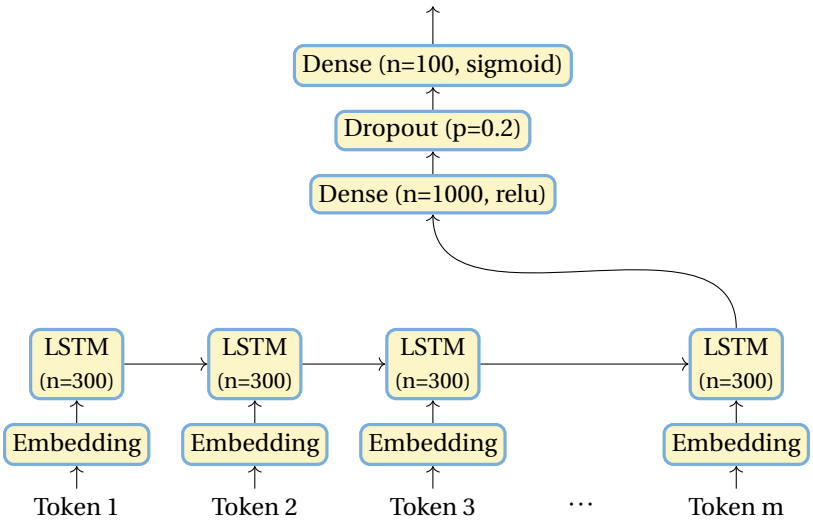


Figure 4.6: Depiction of a one-directional LSTM. The values of n denote the output dimension of the respective layer.

For the Electra model, we use a similar setup based on the hyperparameters depicted in Table 4.1. The input is altered to include a special `[CLS]` token that represents the start of this sequence and whose activation at the end will be used as the output of the transformer model. Followed by an output unit that consists of two linear layers.

Parameter	**Setting**
Number of attention heads	8
Number of hidden layers	6
Neurons in hidden layer	256
Dropout for attention probabilities	0.1
Activation function in the encoder	gelu
Number of epochs in pre-training	100
Number of epochs in fine-tuning	100
Loss function for pre-training	BCEWithLogits
Loss function for fine-tuning	BCEWithLogits
Optimizer	Adam

Table 4.1: (Hyper-)Parameters of the Electra model.

Both models were implemented based on the pytorch library[1]. The Electra-based model was implemented as part of the ChEB-AI framework[2] that was developed in order to simplify AI-based methods around the ChEBI ontology. The BCEWithLogits-loss is the binary cross-entropy variant using the log-sum-exp-trick discussed in 2.4.9.

4.4 Evaluation

4.4.1 Evaluation of the Traditional Methods and LSTM

My contributions

The following section summarizes the results from our collaboration published in [103]. My contributions to this work were the conceptualization, implementation, training and evaluation of the LSTM-based model and production of the respective results, excluding the class-wise evaluation in Section 4.4.1.3. I also authored the related journal article [103].

We will first focus on the results of the more traditional approaches, namely Logistic Regression, k-Nearest Neighbor, Decision Trees, Ransom Forests, Naive Bayes, Linear Discriminant Analysis and Support Vector Machines. These methods have been trained on the balanced, multi-class $M \times N$-datasets described in Section 4.1.1. It should, however, be noted that not all of these approaches follow the same mechanisms. The Logistic Regression, for example, uses the *one-VS-rest* method that trains one classifier for each class, while others aim for a direct multi-class prediction. We will also compare these results to the first sequence-based approach, the Long Short-Term Memory (LSTM) model. This model has been trained on multi-label variants of the 100×500 and 500×100 datasets. The Electra model was not used when this original research was published, and as later tests showed, its performance would not exceed what was achieved with larger data sets.

The multi-class approach bears a particular problem when considering a set of classes with hierarchical relationships. Predicting a direct parent of the target class is still wrong, from the perspective of the classification task, but not logically. Therefore, we also developed a path-based evaluation method that we use to compare our approaches to the current state-of-the-art that is used as part of ChEBI's development process that we will evaluate in Section 4.4.1.4.

[1] `https://www.pytorch.org`
[2] https://github.com/ChEB-AI/python-chebai

4.4.1.1 Evaluation Results by Problem Size

Due to the structure of ontologies, predictions further down the hierarchy are more valuable, as they are more specific, while more general subsumptions follow from the transitivity of the subsumption relation. But this very property also entails that classes that reside on lower levels of the ontology have fewer annotated subclasses that can be used to derive the intended meaning of a class. At the same time, the total number of classes that are higher up in the hierarchy is inherently small. So there is a high number of classes with few annotated leaf-classes and a low number of classes with many annotated leaf-classes and a choice on one value dictates the available range for the other. There are, for example, not 500 classes with 500 members in ChEBI. In this evaluation, we aim to analyze different combinations of these values and their impact on the predictive quality of our models. In general, we would expect that a smaller number of classes, and a larger set of examples to learn from, would yield an easier task for most automated approaches to classification.

Because the LSTM model is a deep-learning model that relies heavily on large amounts of heterogeneous data and is also one of the approaches that required the largest amount of time and resources during training, we have included LSTMs only in selected parts of this analysis.

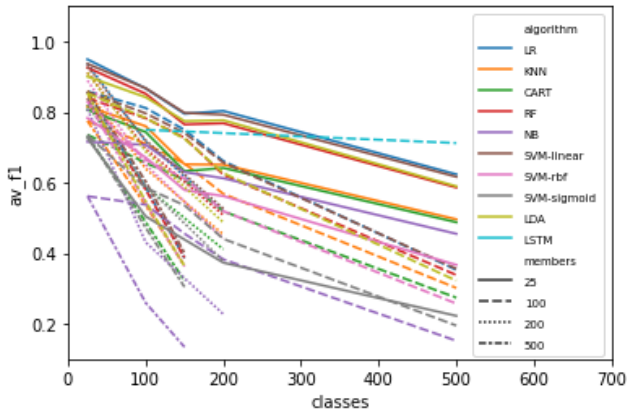
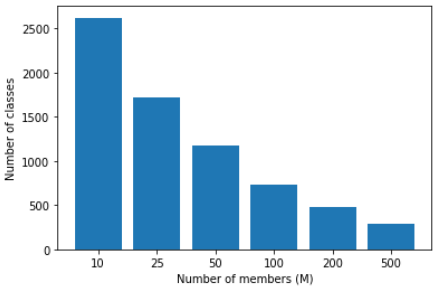


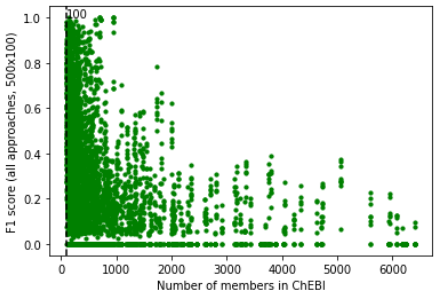
Figure 4.7: Mean F1 score across all classes for different problem sizes. LR=logistic regression; KNN=K-nearest neighbours; CART=decision tree; RF=random forest; NB=naive bayes; SVM=support vector machine; LDA=linear discriminant analysis; LSTM=long short-term memory network.

Figure 4.7 depicts the result of traditional approaches and the LSTM models on different numbers of classes and members. Our analysis shows for the traditional approaches an increasing number of either classes or members per class has a negative impact on the overall performance. While there are some classifiers that performed better on datasets with 100 classes than on one with

25, there is a general steep deterioration once the number of classes increases above 100. As shown in Figure 4.8a, there are less than 1,000 classes that contain at least 100 members. Even the highest number of classes in our dataset (500) therefore only covers a fraction of the ChEBI ontology. This indicates that the extensibility of these approaches to a larger fragment of ChEBI is limited.



(a) Number of classes with at least M members, for different sizes of M, in ChEBI.



(b) Performance of all models in the 500x100 problem for classes with different sizes. Each dot represents the F1 score for a single class in a single model.

Figure 4.8: Class distribution and model performances.

The drop in performance for larger numbers of members per class does not follow the expected behavior of many modern machine learning models. These approaches usually improve performance with a growing amount of data. This behavior may be attributed to different factors. First, the distribution of structural features is not homogenous. Classes that are high up in the hierarchy are often specified by simpler, easy-to-detect concepts. The presence of atoms from the p-block of the periodic table is, for example, the defining feature for *p-block molecular entity* (CHEBI:33675). Conversely, classes that are very specific and further down in the taxonomy are often defined by very specific, distinguishing features. Figure 4.9 depicts an example of a class that requires a very specific molecular sub-structure for all of its subclasses. These rare substructures will generate very unique fingerprints, which makes them easier to detect for fingerprint-based approaches. The second factor is the data selection strategy, which we discussed in Section 4.1.1. This approach ensures that the dataset is constructed as a multi-class dataset. In reality, however, molecules belong to several classes. A molecule that, structure-wise, should belong to a certain class while the dataset does not reflect that, will likely be detrimental to a model's performance. The larger the overlap in classes, the higher the impact of this problem. We have also seen earlier that the ChEBI ontology has a complex subclass hierarchy with overlapping and interleaving branches. Classes with a lower number of members

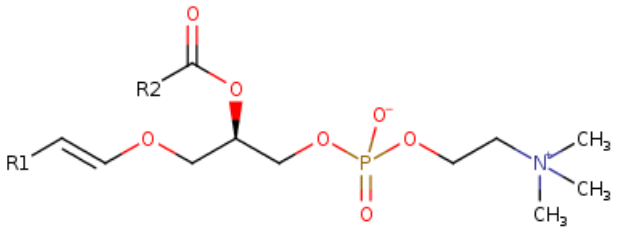


Figure 4.9: (Incomplete) structural specification for *1-O-(alk-1-enyl)-2-O-acyl-sn-glycero-3-phosphocholine* (CHEBI:17810)

generally reside further down in the hierarchy than those with more members. Consequently, for a smaller number of required members, the individual subbranches can be more "spread out" and thereby reduce the potential overlap. A more general class in ChEBI will therefore perform worse in this problem statement, which is also indicated by the negative correlation between F1-score and total member size seen in Figure 4.8b.

This highlights that the problem of chemical classification should be seen as a multi-label problem. We therefore also explored whether multi-label hierarchical classification approaches could mitigate the shortcomings of the classical classifier algorithms applied to this problem. Any of the above classifiers can be used together with a hierarchical classification strategy [111]. In a hierarchical classifier, subsumption relations between the target classes are taken into consideration by training separate classifiers for each of the higher-level nodes in the hierarchy. These then derive predictions just for the levels beneath them. The result is a chain of nested classifiers that are iteratively applied until a leaf node is reached. This is also closely related to the approach that was taken in the natural products classifier mentioned above [78], as in that work a different classifier was trained for each of their three hierarchical levels. We thus evaluated a hierarchical classification approach based on subsets of ChEBI corresponding to the hierarchy above a given set of selected classes. However, we found that this approach in practice did not scale to sub-

sets of ChEBI classes at the problem sizes we have used. This is likely because the complex subsumption hierarchy of ChEBI still requires a large number of classifiers to be queried for a given prediction. Moreover, performing hierarchical classification in the case of ChEBI classes would involve significant redundancy because the classes at the intermediary levels have so much mutual overlap in terms of their lower-level members. Artificial neural network-based approaches can learn hierarchical structures directly, as we will see, thus, we did not further explore hierarchical classifiers at this stage, although we may return to this in future work.

4.4.1.2 Comparison of Different Algorithmic Approaches

Among the classical classifiers, we see that logistic regression performs best (Figure 4.10), followed by linear discriminant analysis, SVMs and random forests performing about the same.

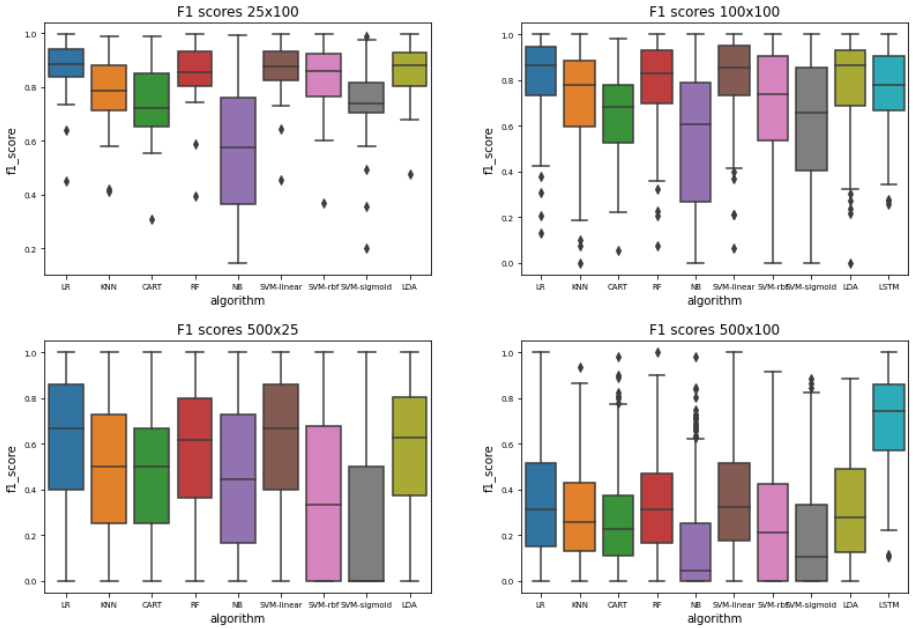


Figure 4.10: F1 scores per algorithm for the 25x100 problem, 100x100 problem, 500x25 problem and 500x100 problem. LR=Logistic regression;KNN=K-nearest neighbours; CART=Decision tree; RF=Random forest; NB=Naive Bayes; SVM=Support vector machine; LDA=Linear discriminant analysis; LSTM=Long short-term memory network

Naive Bayes shows the worst performance among the classical classifiers for this problem. The large difference in performance between Naive Bayes and LDA implies that there are considerable co-variances among the fingerprint features – which would be expected – and can be confirmed by a correlation analysis. These may originate from the fact that the walks that produce the fingerprints are performed on the same substructure, or by the way the hashes

are calculated. Decision trees seem to over-fit – especially with larger sample sizes. The random forests mitigate this to some extent, but a decline in decision tree performance impacts the random forests as well.

The performance of Logistic Regression and Support Vector Machines with a linear kernel is almost identical, which is to be expected as they use essentially the same classification method. The different loss functions used in the respective gradient descents do not have any significant impact. However, non-linear kernels had a highly negative impact on the SVM classification performance.

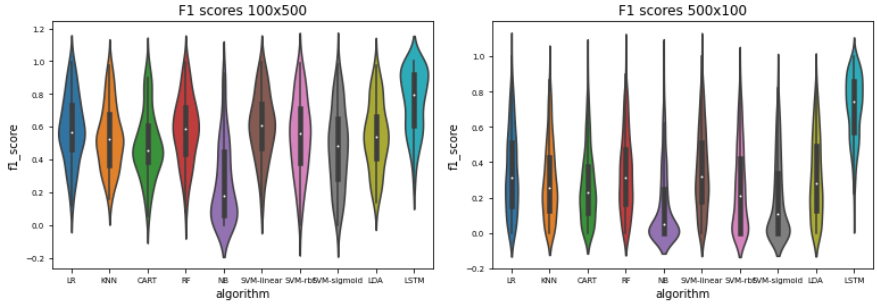


Figure 4.11: Violin plot of F1 scores per algorithm for the 100x500 problem (left) and the 500x100 problem (right). LR=Logistic regression;KNN=K-nearest neighbours; CART=Decision tree; RF=Random forest; NB=Naive Bayes; SVM=Support vector machine; LDA=Linear discriminant analysis; LSTM-Long short-term memory network. Value ranges are not clipped to possible range $[0,1]$, the density function therefore extends the plot outside of the possible range.

While a direct comparison should be interpreted with caution, as the LSTM is performing a different classification task to the other classifiers (i.e. multi-label rather than single-label), nevertheless, we can make some observations about the resulting F1 scores. Interestingly, we see that from the overall performance perspective, although the LSTM does not outperform the other approaches at problem size 100x100 (Figure 4.10), it performs somewhat better at the 100x500 problem size, and significantly better than the other approaches for the 500x100 problem size (Figure 4.11). This implies that the LSTM is, at least on the face of it, better able to scale towards the scope of the full ontology than the classical approaches, although we did not attempt to use the LSTM for problem size categories involving small numbers of members per class (e.g. with 25 members), as network performance decreased with decreased numbers of members per class, as would be expected for this type of approach.

The LSTM networks have been trained on the above datasets for 100 epochs with binary cross-entropy as their loss function. Figures 4.12a to 4.12f show

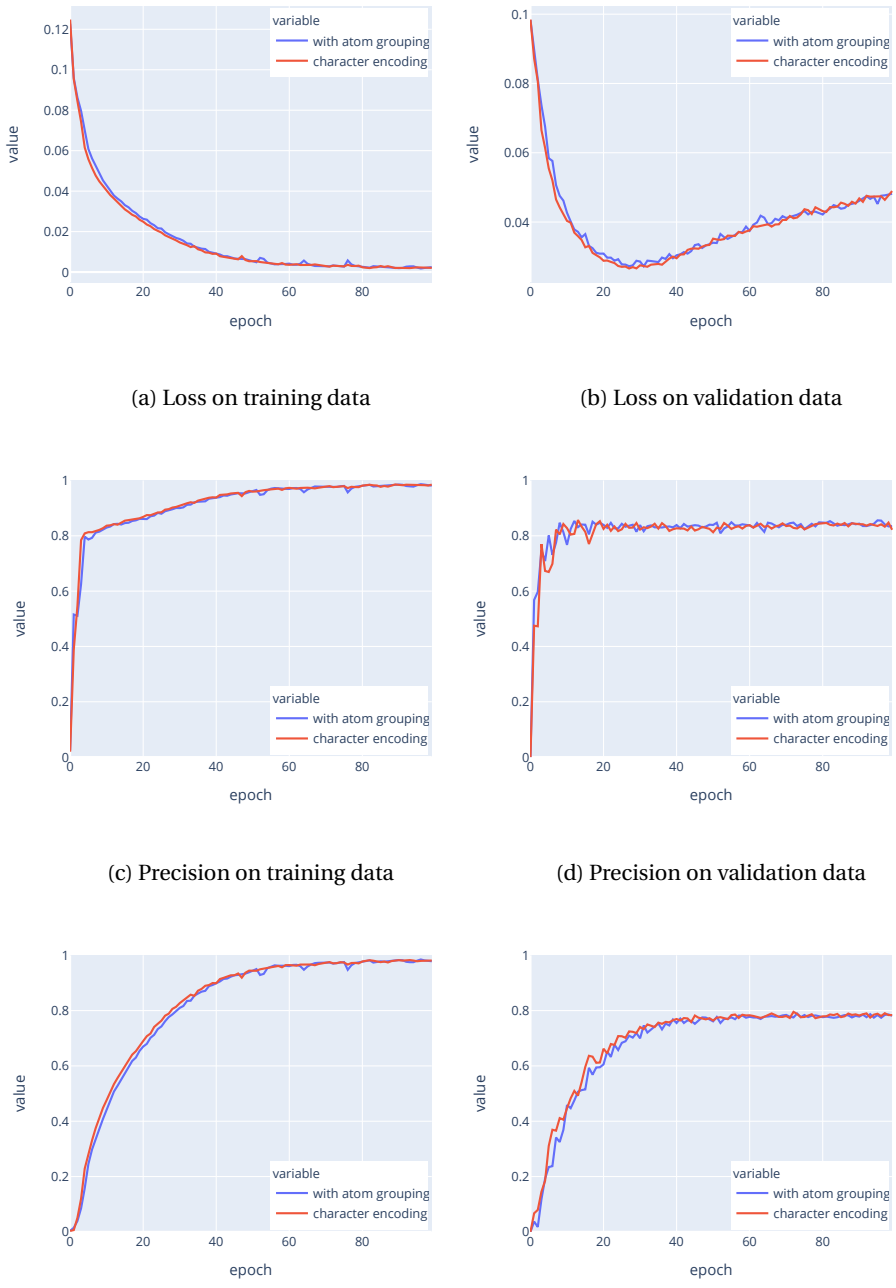
the progress of different metrics during this process. The loss on the validation set rebounds after the 25th epoch, which indicates overfitting on the dataset. Surprisingly, this does not impact the precision and recall negatively. For computing precision and recall, we used a threshold of 0.5 to distinguish class membership from non-membership. Further inspection of the predictions reveals that the mentioned lack of impact is caused by the predictions diverging from the optimal answers towards the threshold, but not passing it. This means that after the turn, prediction strength decreases since distance from the threshold can be seen as confidence about the prediction. The slight, but constant rise of precision and recall after that indicates an additional improvement of those, but apparently at the cost of overfitting.

The encoding of chemicals does not have any significant impact on the success of the learning task. This implies that the networks successfully learn the structure of atom labeling in SMILES strings relatively early on without much effort. Similar experiments in natural language processing [112] have been conducted and results implied that aggregations of syntactic structures have an impact on the training. Our results indicate that this impact does not exist with the tokenizers used in this work.

LSTMs were the approach that did not suffer greatly from larger sample sizes. One aspect of this is that the larger sample sizes create problem spaces that are more uneven, which the LSTM is better suited to handle, as the LSTM is able to make a multi-label prediction and predict multiple classes simultaneously, rather than (as is the case for the other approaches) making just a single prediction. Furthermore, as described above, the data sampling procedure from the ontology will lead to more generic classes if the number of members is larger. This implies that smaller substructures are relevant for the classification, which may be distributed widely across the actual molecules. A random walk has a lower probability of covering all the relevant aspects in this case. The LSTM consumes the whole SMILES string, which allows a more consistent classification.

Figure 4.11 shows that there is a large variance in performance w.r.t different chemical classes. A more class-focused analysis of the results is done in the following section.

It should be noted that we explored several configurations of LSTMs, and none of them performed better than the given configuration, whilst a substantial number showed almost identical results on the validation set. The introduction of a dropout led to a clear rise in performance, whilst different LSTM sizes and structures – even bidirectional ones – showed no positive impact.



(a) Loss on training data

(b) Loss on validation data

(c) Precision on training data

(d) Precision on validation data

(e) Recall on training data

(f) Recall on validation data

Figure 4.12: Metrics recorded during training of the LSTM model.

4.4.1.3 By Chemical Class Within the Ontology

As can be seen by the wide distribution of F1-scores for the performances within each of the different problem sizes and algorithmic approaches, there is variance in the performance of learning for different ontology classes. At the same time, we see variance in the performance of different molecules. This prompts us to ask whether there are some general observations that we can derive about the problem of structure-based chemical ontology classification from these experiments.

Firstly, we can ask whether different algorithms give the same best-performing classes or different best-performing classes. Figure 4.13 shows the overlap of top-performing classes for different problem sizes for the three best-performing algorithms.

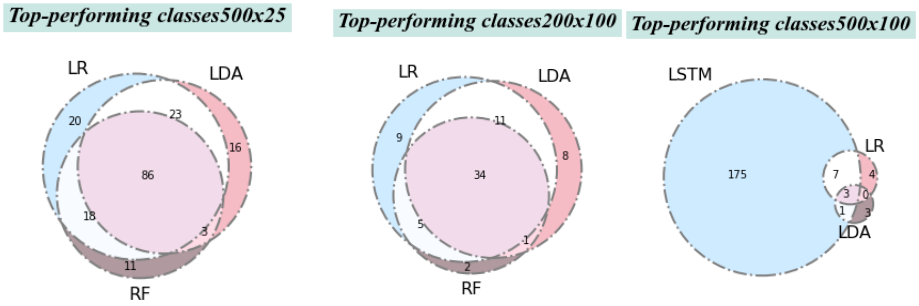


Figure 4.13: The Venn diagrams show the overlap of classes that scored F1 more than 0.8 (i.e., best-performing classes) for three of the classifiers in each of these problem sizes. LR=logistic regression; RF=random forest; LDA=linear discriminant analysis; LSTM=long short-term memory

Figure 4.13 indicates that while there is a shared common core, different classifiers give partially non-overlapping sets of 'best scoring' classes. That is, they have partially distinct profiles with respect to the classes for which they give the best performance. This suggests that the general problem of structure-based chemical ontology classification might benefit from ensemble-based approaches that integrate results across different models.

In general, the classical classifiers perform best on classes that have relatively few members and well-defined structural features. For example, the best performers (F1=1.0) using the LR algorithm all have fewer than 50 members (Table 4.2).

The worst-performing classes for the classical classifiers, exemplified by the worst-ranked for the LR algorithm indicated in Table 4.2, include those with features not directly represented in the fingerprint, such as D-stereochemistry, cations and salts. The information that would be required to make these classifications is just not available for these classifiers to learn, however, these could be improved with the adoption of alternative fingerprinting strategies.

ChEBI-ID	f1	class name	num. members
CHEBI:61689	1.0	amino cyclitol	41
CHEBI:26253	1.0	polyprenylhydroquinone	45
CHEBI:134209	1.0	aporphine alkaloid	54
CHEBI:132157	1.0	hydroxy-1,4-naphthoquinone	35
CHEBI:17810	1.0	1-O-(alk-1-enyl)-2-O-acyl-sn-glycero...	46
CHEBI:17636	1.0	sphingomyelin d18:1	46
CHEBI:26255	1.0	prenylquinone	38
CHEBI:83563	1.0	long-chain alkane	29
CHEBI:60687	1.0	cembrane diterpenoid	35
CHEBI:38836	1.0	1-benzothiophenes	43
CHEBI:36685	1.0	chlorocarboxylic acid	33
CHEBI:75946	1.0	cytochalasan alkaloid	30
CHEBI:38768	1.0	phthalazines	41
CHEBI:80291	1.0	aliphatic nitrile	35
CHEBI:38769	1.0	indazoles	45
CHEBI:37531	1.0	polyprenyl diphosphate	41
CHEBI:58168	1.0	1-O-acyl-sn-glycero-3-phosphocholine	37
CHEBI:83876	1.0	cationic sphingoid	30
CHEBI:64590	1.0	monoalkyl-sn-glycero-3-phosphocholine	30
CHEBI:131903	1.0	pyranopyrazole	32
ChEBI-ID	f1	class name	num. members
CHEBI:16733	0.0	D-alpha-amino acid	51
CHEBI:48544	0.0	methanesulfonates	54
CHEBI:33702	0.0	polyatomic cation	2178
CHEBI:47704	0.0	ammonium salt	38
CHEBI:59869	0.0	L-alpha-amino acid zwitterion	53
CHEBI:50128	0.0	biflavonoid	53
CHEBI:25414	0.0	monoatomic monocation	32
CHEBI:46899	0.0	benzothiazine	37
CHEBI:35218	0.0	anthocyanin cation	47
CHEBI:64985	0.0	bioconjugate	39
CHEBI:33639	0.0	ortho- and peri-fused compound	56
CHEBI:38716	0.0	carboxylic acid dianion	311
CHEBI:59635	0.0	organophosphonate oxoanion	38
CHEBI:35284	0.0	ammonium betaine	1205
CHEBI:29089	0.0	1,2-diacyl-sn-glycerol 3-phosphate	52
CHEBI:35296	0.0	ortho-fused polycyclic arene	38
CHEBI:26469	0.0	quaternary nitrogen compound	1317
CHEBI:38037	0.0	methanesulfonate salt	40
CHEBI:76176	0.0	2-hydroxy fatty acid anion	43
CHEBI:59558	0.0	medium-chain fatty acid anion	36

Table 4.2: Highest and lowest-scoring classes using the LR algorithm

The profile of poor performers is different for the LSTMs compared to the classical approaches.

Table 4.2 shows the twenty best and twenty worst performing classes with the LSTM approach according to their respective F1 score. The best-performing classes also include classes that have well-defined structural features, but these have far more members than the best performers in the LR approach, illustrating the ability of the LSTM to cope with larger problem sizes – and the added value of additional examples to learn from.

The worst-performing classes for the LSTMs have a quite different profile to those of the LRs, and as expected do not include salt or ion classes. Rather, somewhat intriguingly, we see many examples of classes with complex ring structures, especially aromatic or substituted ring structures.

To confirm this observation, we applied the BiNChE chemical enrichment analysis utility [57] on the 50 worst-performing classes from the LSTM-only set, we see a number of clear enrichments – benzenes, aromatic compounds, and carbocyclic compounds (Figure 4.14), while in the worst-performing classes from all algorithms we see no similar enrichment.

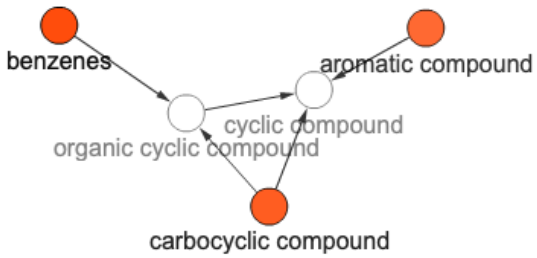


Figure 4.14: Enrichment analysis result on the ChEBI structural ontology for the 50 worst-performing classes in the LSTM

We can hypothesize that the poor performance for the aromatic molecules with the LSTM may be due to the fact that aromaticity can be encoded in SMILES strings in multiple different ways – using alternating single and double bonds or using lowercase letters. It is plausible that the network did not learn that e.g. the aromatic 'c' carbon atom is in fact the same atom type as the typical 'C' in another molecular representation, and treated them as different entities. Larger datasets from possibly synthetic sources or a more homogenous representation of aromatic components may help the network to learn those abstractions. It is also worth observing that in general the LSTM can be expected to have more difficulty with parsing cycles than linear molec-

ChEBI-ID	f1	class name	num. members
CHEBI:17984	1.000000	acyl-CoA	696
CHEBI:37240	0.998350	adenosine 3',5'-bisphosphate	697
CHEBI:22251	0.995114	adenosine bisphosphate	702
CHEBI:61078	0.993104	purine nucleoside bisphosphate	706
CHEBI:58946	0.992382	acyl-CoA oxoanion	707
CHEBI:61079	0.991522	ribonucleoside bisphosphate	707
CHEBI:51277	0.990164	thioester	745
CHEBI:37123	0.989925	nucleoside bisphosphate	708
CHEBI:60971	0.989796	aminophospholipid	104
CHEBI:18303	0.989796	phosphatidyl-L-serine	104
CHEBI:64583	0.986667	sphingomyelin	251
CHEBI:58342	0.985302	acyl-CoA(4-)	613
CHEBI:74927	0.985222	furopyran	934
CHEBI:35766	0.983871	glycerophosphoserine	129
CHEBI:78799	0.983607	hydroxy fatty acid ascaroside	152
CHEBI:52565	0.980769	acylglycerophosphoserine	114
CHEBI:26875	0.980392	terpenyl phosphate	133
CHEBI:36233	0.980392	disaccharide	156
CHEBI:64482	0.979315	phosphatidylcholine	623
CHEBI:57643	0.979315	1,2-diacyl-sn-glycero-3-phosphocholine	621
ChEBI-ID	f1	class name	num. members
CHEBI:64365	0.333333	aralkylamino compound	138
CHEBI:22715	0.333333	benzimidazoles	352
CHEBI:23697	0.328358	dichlorobenzene	452
CHEBI:48470	0.322581	amidobenzoic acid	148
CHEBI:25235	0.320000	monomethoxybenzene	247
CHEBI:46848	0.315789	N-arylpiperazine	176
CHEBI:51681	0.305085	dimethoxybenzene	269
CHEBI:26455	0.294118	pyrroles	200
CHEBI:83403	0.293333	monochlorobenzenes	429
CHEBI:50995	0.292683	secondary amino compound	417
CHEBI:27024	0.277778	toluenes	135
CHEBI:37407	0.259887	cyclic ether	806
CHEBI:73539	0.256410	naphthyridine derivative	162
CHEBI:26878	0.251429	tertiary alcohol	756
CHEBI:36786	0.246246	tetralins	108
CHEBI:27116	0.235294	trihydroxyflavone	150
CHEBI:38338	0.235294	aminopyrimidine	149
CHEBI:33572	0.222222	resorcinols	153
CHEBI:83812	0.114286	non-proteinogenic amino acid derivative	145
CHEBI:13248	0.108108	anilide	279

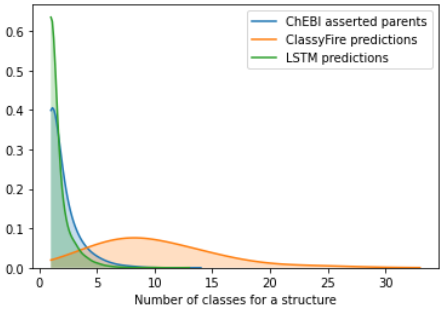
Table 4.3: Highest and lowest-scoring classes using the LSTM algorithm

ular structures from SMILES strings because these structures are broken up during the translation of a molecule into its SMILES representation.

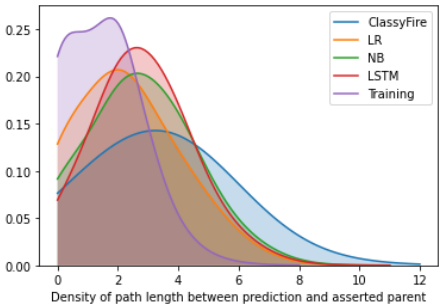
4.4.1.4 Comparison to the State of the Art

As a final evaluation, we compare our results to the state-of-the-art structure-based ontology classification tool, ClassyFire [25]. We do this comparison using as input the 500x100 problem size dataset, by executing ClassyFire on the SMILES strings associated with the test set of molecules, encompassing 20% of the full set of 50,000 molecules, i.e. 10,000 sample molecules with SMILES. Of these, ClassyFire was unable to process 501 of them due to errors in the generation of an InChI (IUPAC international chemical identifier, [113]) from the SMILES. ClassyFire uses an InChI-Key-indexed cache of parent classes for molecules that have been previously classified in order to speed up its classification performance, as matching multiple substructural patterns is expensive. Certain structures cannot be represented by InCHI but are expressible in SMILES, such as octahedral and square planar geometries [114] or structures with undefined substructures or attachment points. These 501 molecules are of this type – mainly due to the explicit representation of attachment points within the SMILES, e.g. the following SMILES: `C(C(COP(=O)(OC[C@@H](C(=O)O)N)O)OC(=O)*)OC(=O)*`. There were also a few entries for which ClassyFire returned other errors. In total, we received 9,484 classification results for our 10,000 sample molecules. Each classification result includes multiple ChEBI classes including the very high-level ChEBI classes such as 'molecular entity'. We condensed these to only include classes that were not superclasses of each other.

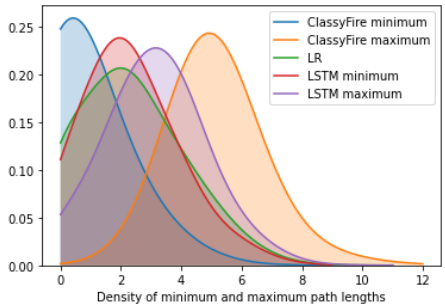
It is not straightforward to make a direct comparison between our results and the performance of the ClassyFire tool, for various reasons. First, ClassyFire uses a different underlying ontology to ChEBI that is only partially mapped to ChEBI. The ontologies differ in some fundamental ways in their treatment of chemical classes. For example, ClassyFire's classes include molecules with different charge states, encompassing conjugate bases and acids in the same grouping, while ChEBI strictly separates these. Therefore, ChEBI class predictions returned by ClassyFire may be less precise than the ClassyFire original class. However, our dataset is restricted to the ChEBI classification from which it was generated. Second, ClassyFire makes multiple parent class predictions, while our classical classifiers make only a single best match parent class prediction, and although the LSTM is able to make multiple predictions, it makes far fewer predictions than ClassyFire does. Figure 4.15a shows a kernel density diagram for the number of parent classes in different approaches: 1) ChEBI directly asserted parents (with a mean of 1.816 parent classes per leaf structure



(a) Number of parent classes in different approaches.



(b) Path length to asserted parent class in different approaches.



(c) Minimum and maximum path lengths to asserted parent classes.

Figure 4.15: Analysis of path length between ClassyFire and ChEBI

across the full ontology), 2) the LSTM predicted parent classes (mean = 1.435 in the 500x100 problem), and 3) ClassyFire predicted parent classes (mean = 9.926). For both ClassyFire and the LSTM, these counts exclude any parent classes returned by the algorithm that are superclasses of any of the other parent classes. The higher number of classes predicted by ClassyFire is, however not very useful for domain experts because these classes are too general. The bottom-up nature of our data selection algorithm (Section 4.1.1) terminated before these classes were reached. Finally and most importantly, ClassyFire has since its initial release in 2016 been used in the development of ChEBI: it is used in the bulk submissions pipeline to automatically classify entities that are incorporated into ChEBI before they can be manually curated. This means that ClassyFire has actually produced a portion of the classifications in our dataset (both training and test), although these are not flagged or indicated as such in any way. This introduces a bias that is difficult to fully address.

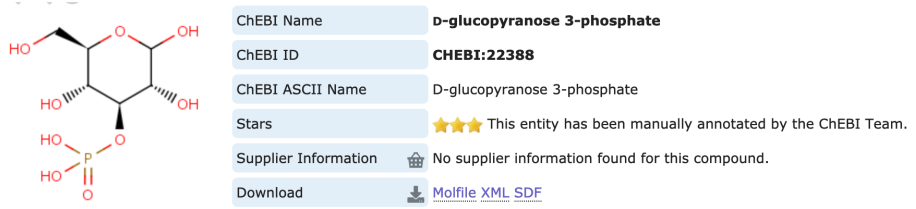


Figure 4.16: D-glucopyranose 3-phosphate, an example molecule for which ClassyFire performs poorly on this metric.

We compare the approaches by computing a path length distance between what we might call the 'ground truth' of the asserted classification in ChEBI, and the predicted classification. That is, we count the number of subclass relations that must be traversed to get from a directly asserted parent to the predicted parent. In practice, the longer paths tend to reflect classifications that are either wrong (in a different ontology branch) or not very useful (at a very high level). Thus, path length provides a useful metric for the quality of a classification. As there may be multiple directly asserted parents and multiple predicted parents, for each structure in the test set, we computed all path lengths

between pairwise combinations of asserted parents and predicted parents. If the predicted class was identical to one of the asserted parents, we added a path length of 0 to indicate a match. Note that the asserted parents in ChEBI are not always the class that we used as input to our classifiers, due to the selection processing of the ontology for learning purposes. Thus, we compute the path lengths also on the classes that we used as the selection. Figure 4.15b illustrates the overall density of the returned path lengths in this metric, with the selected classes indicated as 'Training', showing the results for the LSTM,

the best and worst of the classical approaches, and ClassyFire. It can be seen that ClassyFire returns the widest range of path lengths on this metric with a mean path length of 3.20, while the LR (mean=2.29) outperforms the NB (mean=2.74) and the LSTM (mean=2.81), which appear to perform similarly. The training baseline for our learning approaches has mean=1.48.

These results may reflect a bias based only on the number of paths computed. For that reason, we calculated also the minimum path length and the maximum path length (Figure 4.15c). The minimum path length in particular addresses an important limitation of measuring the path lengths: Given two target classes and two correct predictions of a molecule as a member of these classes, the path lengths would still include the path length between these two classes, even though the predictions were correct. The minimum path length would be 0 in this case. On the minimum path length, ClassyFire outperforms the other approaches, while on the maximum path length, it shows the worst performance. However, in practice for a novel structure, it would not be known without manual inspection which of the results returned was the best classification – reducing the benefit of using an automated approach. To illustrate why the maximum path length of ClassyFire is significantly longer than for the other approaches, let's consider the molecule 'D-glucopyranose 3-phosphate' as an example. In ChEBI it is classified as glucose phosphate that is a derivative of hexose. ClassyFire returns the following predicted classifications for this molecule: 'primary alcohol' (CHEBI:15734), 'secondary alcohol' (CHEBI:35681), 'ether' (CHEBI:25698), 'monoalkyl phosphate' (CHEBI:25381), 'polyol' (CHEBI:26191), 'oxanes' (CHEBI:46942), 'hemiacetal' (CHEBI:5653), 'hexose' (CHEBI:18133), and 'organic oxide' (CHEBI:25701). Many of these classifications relate to correct but very general chemical groupings , illustrating the challenges with the substructure-based approach to automated structure-based ontology classification in the context of the large and combinatorial chemical structural landscape. Whilst they are correct, these classifications are not useful for most use cases and there are more specific classes that should be assigned instead. Due to the different concept distinctions discussed in Section 2.3.3 and therefore loose alignments, ClassyFire is unable to predict these classes. Other classifications made by ClassyFire are incorrect in ChEBI, often due to differences between ClassyFire and ChEBI's approach to classification (e.g. hexose vs. hexose derivative). It may be beneficial to introduce an additional penalty to these predictions. Defining such a penalty is, however, not trivial because cases in which ClassyFire's predictions contradict classifications in ChEBI may also originate from parts of ChEBI's taxonomy, that are not well structured. The class of *peptide* (ChEBI:16670), has a large number of subclasses that should be included as subclasses of

their siblings. Defining a correctness-based penalty would therefore require manual examination of ClassyFire's predictions.

ClassyFire has already had an enormous impact on the field of chemical data management by enabling novel structures to be classified. However, our results underline that there is a role for dynamic machine learning-based approaches alongside substructure-based approaches. ClassyFire predicts for each molecule significantly more (non-redundant) parent classes (mean = 9.926) than the LSTM (mean = 1.435) or the LR (1 prediction), where - roughly speaking - often, one of these predicted parent classes is better than the classes predicted by the LSTM or the LR in our path distance metric (bearing in mind the possible bias due to the use of ClassyFire in ChEBI development), but most of them are worse. ClassyFire seems particularly suitable for semi-automatic use cases, where the results that are returned by ClassyFire are validated manually. However, if manual validation is not possible, the other approaches appear to be more suitable, in particular, if they can be used together in a way that plays to the differential strengths of the different approaches. Importantly, they are also likely to be easier to maintain and extend going forward.

4.4.2 Evaluation of the Electra-based Models

My contributions

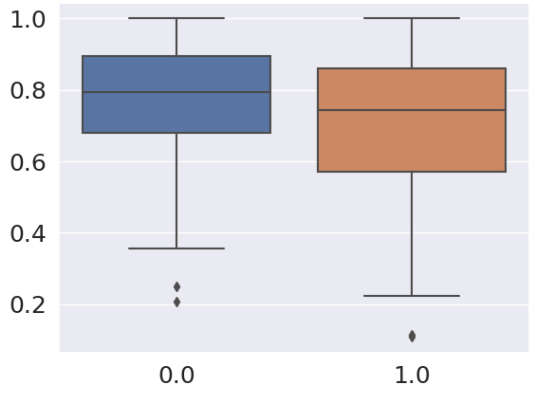
The following section summarizes the results from our collaboration published in [104, 115]. My contributions to this work were the conceptualization and implementation, pre-training, training and evaluation of the predictive quality of the Electra-based models on the $\text{ChEBI}^{500*}_{v200}$, $\text{ChEBI}^{500}_{v200}$ and $\text{ChEBI}^{854}_{v200}$ dataset. This does not include the classification of never-seen chemicals in Section 4.4.2.2. Furthermore, I was the main author of the related book chapter [106] and journal article [104]. I also supervised a master's thesis and authored the subsequent paper [105] on a related investigation using the ChemBERTa [116] model.

For the evaluation of the Electra-based models, we employed five different datasets ($\text{ChEBI}^{500*}_{v200}$, $\text{ChEBI}^{500}_{v200}$, $\text{ChEBI}^{854}_{v200}$, $\text{ChEBI}^{1332}_{v200}$, $\text{ChEBI}^{709}_{v148}$) that emerged from our initial analysis of traditional approaches discussed in Section 4.4.1. Following this iterative process, we first compared the performance of the Electra-based model with the LSTM model which was the best-performing model for wide-range ontology extension, as shown in our previous analysis. Table 4.4 compares the results of the current model with the previously obtained results for the LSTM model on the $\text{ChEBI}^{500*}_{v200}$ dataset

using the Chem tokenizer. The Electra-based model outperformed the LSTM-based model in both micro and macro aggregations. The macro-aggregation is of particular importance for the task of ontology extension. Classes that are higher up in the taxonomical hierarchy are more likely to have more members in the dataset than those lower down. The micro-aggregation weights the performance of these classes equally, while a poor performance for a single class has a higher impact on the macro-F1 score.

	Macro		**Micro**	
	LSTM	**ELECTRA**	**LSTM**	**ELECTRA**
F1	0.71	**0.79**	0.74	**0.79**
Recall	0.68	**0.76**	0.70	**0.77**
Precision	0.77	**0.80**	0.79	**0.82**

Table 4.4: The comparison of the scores achieved by two models on the ChEBI$^{500*}_{v200}$ dataset. Better values are represented in bold font.



(c)

Figure 4.17: Boxplots for the F1 scores of all 500 classes on the ChEBI$^{500*}_{v200}$ dataset. A statistical test comparing the two class-wise F1 scores distributions yields a p-value of less than 0.001, indicating the distributions significantly differ and that Electra (blue) outperforms the LSTM model (red).

Due to the improved performance over the previous approaches, we selected the Electra-based model for further analyses. We discussed in Section 4.1.2 that the ChEBI$^{500*}_{v200}$ dataset did not include all possible members of a given class. Therefore, we created the ChEBI$^{500}_{v200}$ dataset that contained all these additional instances and fine-tuned an additional Electra-based model on this new dataset.

	Macro		Micro	
	$\text{ChEBI}^{500*}_{v200}$	$\text{ChEBI}^{500}_{v200}$	$\text{ChEBI}^{500*}_{v200}$	$\text{ChEBI}^{500}_{v200}$
F1	**0.79**	0.77	0.79	**0.82**
Recall	**0.76**	0.74	0.77	**0.79**
Precision	**0.80**	**0.80**	0.82	**0.84**

Table 4.5: Comparison of the scores achieved by the Electra-based model with Chemtokenizer on the $\text{ChEBI}^{500*}_{v200}$ and $\text{ChEBI}^{500}_{v200}$ dataset after being pretrained on *Mol-Pretrain*-dataset based on different embeddings.

Table 4.5 shows the different F1-scores for the Electra model based on the $\text{ChEBI}^{500*}_{v200}$ dataset and the $\text{ChEBI}^{500}_{v200}$ dataset. The models perform differently based on the aggregation method used. While the model produced higher macro scores on the smaller set than on the larger one, the opposite is the case for the micro aggregation. We attribute this to the different distribution of class sizes in both datasets. While the design for the original $\text{ChEBI}^{500*}_{v200}$ dataset was built in a way that focused on a balanced distribution of members in classes, we lifted this restriction entirely to achieve a larger amount of training data. Consequently, the classes in the extended dataset contain more instances and due to the transitive property of the subsumption relation, classes higher up in the hierarchy have more annotated subclasses than those further down in the hierarchy. The training loss, however, is calculated instance-wise. The training on the extended dataset, therefore, favors those classes that have a higher number of instances. Nevertheless, it shall be noted that the data in Table 4.5 should not be used to conclude that one dataset is better than the other as they express very different data distributions. The extended dataset gives a better representation of the way that domain ontologies are structured and the larger amount of training data hedges the model against out-of-distribution errors.

	Macro		Micro	
	BPE	**Chemical tokens**	**BPE**	**Chemical tokens**
F1	0.75	**0.77**	**0.84**	0.82
Recall	**0.75**	0.74	**0.86**	0.79
Precision	0.75	**0.80**	0.82	**0.84**

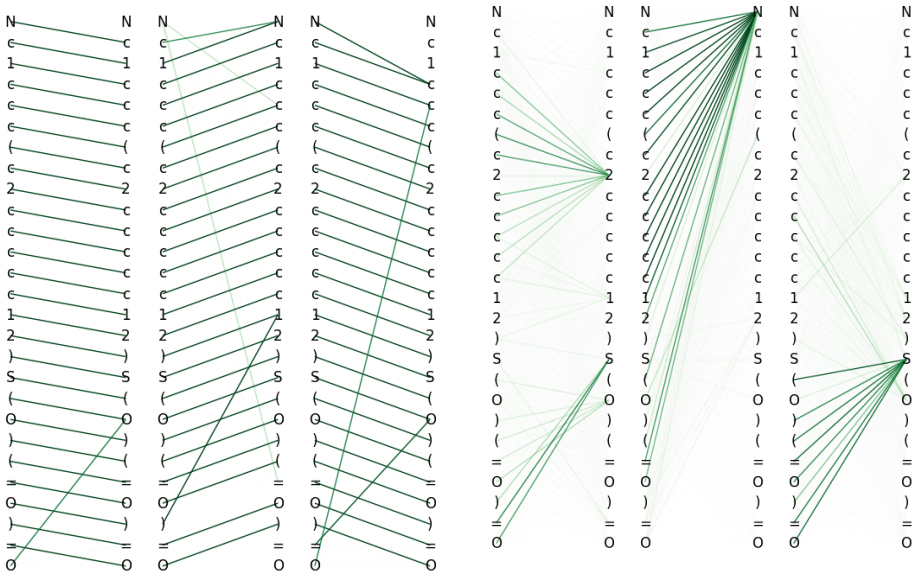
Table 4.6: Comparison of the scores achieved by Electra on the $\text{ChEBI}^{500}_{v200}$ dataset after being pretrained on *Mol-Pretrain*-dataset based on different tokenization methods.

Another aspect we were interested in was the impact of different tokenization strategies on the Electra-based models. Therefore, we pre-trained and fine-tuned an additional Electra-based model using the BPE-tokenizer and compared it with the version using the chemistry-oriented *Chem* tokenizer. The results are shown in Table 4.6. Similar to what we have seen when we analyzed the behavior in different datasets, both methods appear better with respect to different aggregation methods. In this case, the model using the *BPE*-tokenizer outperformed the one using the *Chem*-tokenizer when considering the Micro-aggregation. The opposite is true for Macro-aggregation. This may again be attributed to the different distributions of members in the dataset. The molecular substructures that a model needs to detect in order to correctly classify a member class are much simpler for more general classes. Those further down in the hierarchy require more complex structures and, as we discussed in Section 4.2, BPE tends to merge tokens in ways that are not chemically meaningful. This was particularly true for numeric cycle identifiers. These structures are also more prominently present in more specific classes. The slight improvement in Micro-F1-score, however, comes at the expense of interpretability because the frequency-based aggregation results in chemically infeasible tokens. We will therefore limit further discussions to the chemical tokenization. The macro aggregation is also more relevant for the $\text{ChEBI}^{500}_{v200}$ dataset, as the inclusion of additional instances increased the imbalance.

4.4.2.1 Interpretability

One of the big advantages of transformer-based models is not just their predictive power and versatility, but also the embedded attention mechanism that we discussed in Section 2.4.8. This part of the architecture allows the model to learn a binary relation between different kinds of tokens. The combination with positional embeddings further enables relations between different positions of the input sequence. These relationships are then used to filter parts of the input sequence, allowing the model to focus on those parts that are important for the given classification task [117]. The inspection of this learned relationship can, therefore, give important insights into the structures that led to a particular classification [118, 119, 120, 121]. We also described earlier that these attention mechanisms utilize several lower-dimensional modules, the so-called attention heads, that do not share their respective projections M_q, M_k, and M_v. This allows the model to learn multiple filtering criteria in parallel. Our model consisted of 6 layers with 8 attention heads each, which resulted in a total of 48 different attention relations per input sequence.

For the input structures for our Electra model with the Chem tokenizer, this means that we get a fuzzy binary relationship between different atoms, bonds and other syntactic elements of the SMILES representation. We examined two different ways to visualize these relationships. The first one is the straightforward representation of this relationship as done in Figure 4.18 for selected attention heads and layers for the molecule *naphthionic acid* (CHEBI:38219). Darker lines in the network plot indicate a stronger attention relation between tokens. Similarly, darker green in the molecule plot indicates that higher attention was paid to that particular part of the molecule. These plots, however, can be hard to read in particular for those who are not familiar with the SMILES notation. Therefore, we also depict the attention for a specific input token in the query input in Figure 4.19.

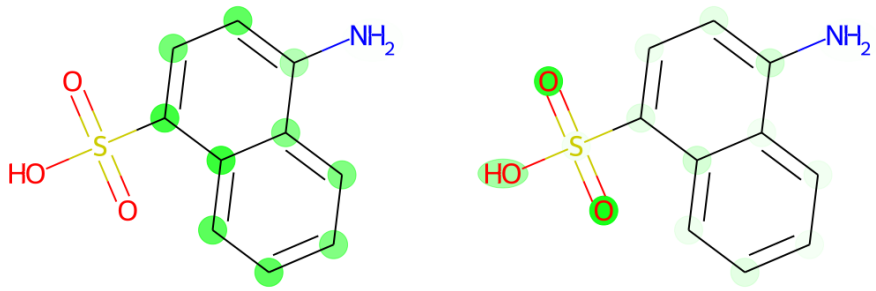


(a) Attention of naphthionic acid (CHEBI:38219) in layer 2, heads 1-3.

(b) Attention of naphthionic acid (CHEBI:38219) in layer 5, heads 4-6.

Figure 4.18: Attention relations for naphthionic acid. Query input is depicted on the right, key input is depicted on the left of each plot.

According to ChEBI, *naphthionic acid* (CHEBI:38219) is a subclass of *arenesulfonic acid* (CHEBI:33555). The latter is defined as a sulfonic acid that features one or more carbon rings. These are also the structures that caused high attention in our model when processing the SMILES string with which the class of *naphthionic acid* (CHEBI:38219) is annotated in ChEBI. The left column of Figure 4.18b illustrates the model's attention from the ring index "2" at position 8 to the fused carbon ring structure. The highlighted portions in Figure 4.19a represent the areas to which the system allocates high attention when exam-



(a) Attention for naphthionic acid in layer 5, head 4, when visiting cycle index marker 2

(b) Attention for naphthionic acid in layer 5, head 6, when visiting sulfur

Figure 4.19: Attentions per input token in the query input for naphthionic acid, projected to the molecule

ining ring index "2." The right column of Figure 4.18b and, correspondingly, Figure 4.19b displays the model's attention to the sulfonic acid group originating from the sulfur atom "S" at position 17. These structural elements play a crucial role in accurately classifying the molecule as an *arenesulfonic acid*.

Visualizations such as those in Figure 4.19 provide a representation of the attention structure that is more intuitive for chemists and enable a sort of visual explanation for the classification. The middle column of attention weights in Figure 4.18b shows that the model also paid attention to a substantial portion of the molecular structure from the nitrogen atom at position 0. Nitrogen is the essential element in amino compounds. As there were other amino classes that were part of the 500 selected classes, such as *amino monosaccharide* (CHEBI:60926), glutamic acid derivative (CHEBI:22693) or ethanolamines (CHEBI:23981), it is possible that the model needs to analyze larger parts of the molecule in order to rule out those possible candidates for superclasses.

We also found that some universal patterns emerged during classification. The attention distribution in the first layer was almost homogenous. The second and third layers focused mostly on direct neighborhoods (preceding and following tokens) within the SMILES string, as depicted in Figure 4.18a. Attention to neighborhood tokens is important for understanding how general chemical connectivity within molecules is specified in the SMILES language. The fourth, fifth and sixth layers focused on larger structures, as shown in Figure 4.18b.

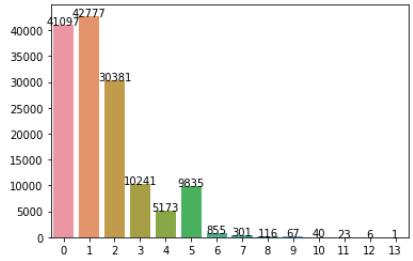
The visualizations that we presented in this section provide a useful way to interpret the attention mechanism of the model, because in the case of true positives or true negatives, they enable a chemist to check whether the classification by the model was indeed based on the chemically salient substructures of a molecule. If that is the case, it provides some validation for the assumption that the model learned some chemically meaningful distinctions. Analogously, if a molecule is classified wrongly because the model fails to pay attention to the presence of one of the relevant substructures (e.g., a false negative classification), then the visualization of the attention mechanism is helpful to understand which of the salient substructures the model fails to recognize. A presentation of these predictions and visualizations to a group of chemists informally received enthusiastic confirmation that these visualizations provide meaningful insights into the operation of the system accessible to domain experts. We, therefore, expect the visualizations presented here to improve the ontology development process. However, our approach is limited to the visualization of attention to the presence of substructures. This is a significant limitation because some chemical classes are defined by the absence of certain structures, which our approach cannot visualize. It is, therefore, not known to us, whether the Electra model uses the absence of structures for classification.

4.4.2.2 Classification of never-seen chemicals

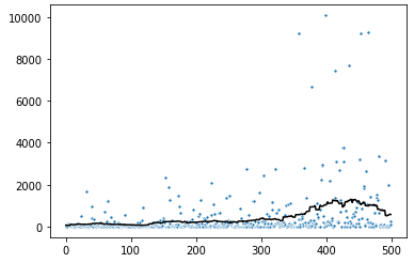
To evaluate the performance of the model on never-seen chemicals, we applied it to a set of 140,913 SMILES extracted from PubChem that had 'hazard class' annotations and were not present in ChEBI.

Among these, we found that 29% (41,097) – almost a third – of the input molecules were not assigned any class at all (Figure 4.20, left), which can be regarded as a 'don't know' response from the model. Assessing these molecules reveals that many contain isotopic SMILES with unusual atoms such as 13C Carbon isotopes or 2H Hydrogen isotopes, not present in the training dataset. Others had explicit charges, complex branching structures and explicit stereochemistry, all attributes that are poorly represented in the training data. Although a high percentage, this result is consistent with our earlier findings with similar models and offers a good motivation to connect deep learning-based models together with other approaches in ensembles so that if the learning-based model offers a 'don't know' response, the system as a whole can still offer a reasonable prediction for every input molecule.

The remaining input molecules were assigned to at least one class, and amongst those 57,039 were assigned to multiple classes, including one molecule that was assigned to 13 distinct classes (Figure 4.20, left). The distribution of molecules into classes is spread quite well across the 500 classes with which



(a) Bar plot indicating how many molecules were assigned a particular number of class predictions, for example, the bar at 0 indicates that 41,097 molecules were assigned no class prediction.



(b) Scatter plot indicating the number of molecules predicted to belong to each of the 500 ChEBI classes in the prediction task (ordered), with the rolling average indicated by a line.

Figure 4.20: Plots depicting the distribution of predictions on the hazardous dataset.

the model is trained (Figure 4.20, right). Figure 4.21 shows a subset of the ChEBI hierarchy illustrating the assignment of molecules to classes, with colors indicating how many molecules were assigned to each class (red=fewer, green=more). In general, the classes lower down in the hierarchy received fewer molecule assignments, as would be expected. The full visualization of all the 500 selected classes is available at Zenodo[3].

We then created an ontology extension from the subset of ChEBI classes included in our model with newly added classes based on the classified input molecules. Each input molecule is assigned a class in the extended ontology and the predicted classes are asserted as superclasses. The extended ontology is available at Zenodo[4].

We combined this extended ontology with the disjointness axioms between chemical classes in ChEBI [122] and then tested for consistency using the HermiT reasoner in Protégé. Most of the predicted classifications were consistent, but we found that 5 of the input molecules' predicted classifications resulted in a disjointness violation. These were caused by:

- Classification as both carbohydrate acid and carbohydrate acid 'derivative'. In these cases, the derivative class shares the same core structural features as the class from which it is derived; however, the class definition reflects not only the presence but also the absence of features, which may be harder to learn.

- Classifications into multiple parent classes involving specific counts of groups, e.g. as both a diglyceride and a triglyceride. In these cases, per-

[3] https://doi.org/10.5281/zenodo.6023497
[4] https://doi.org/10.5281/zenodo.6023497

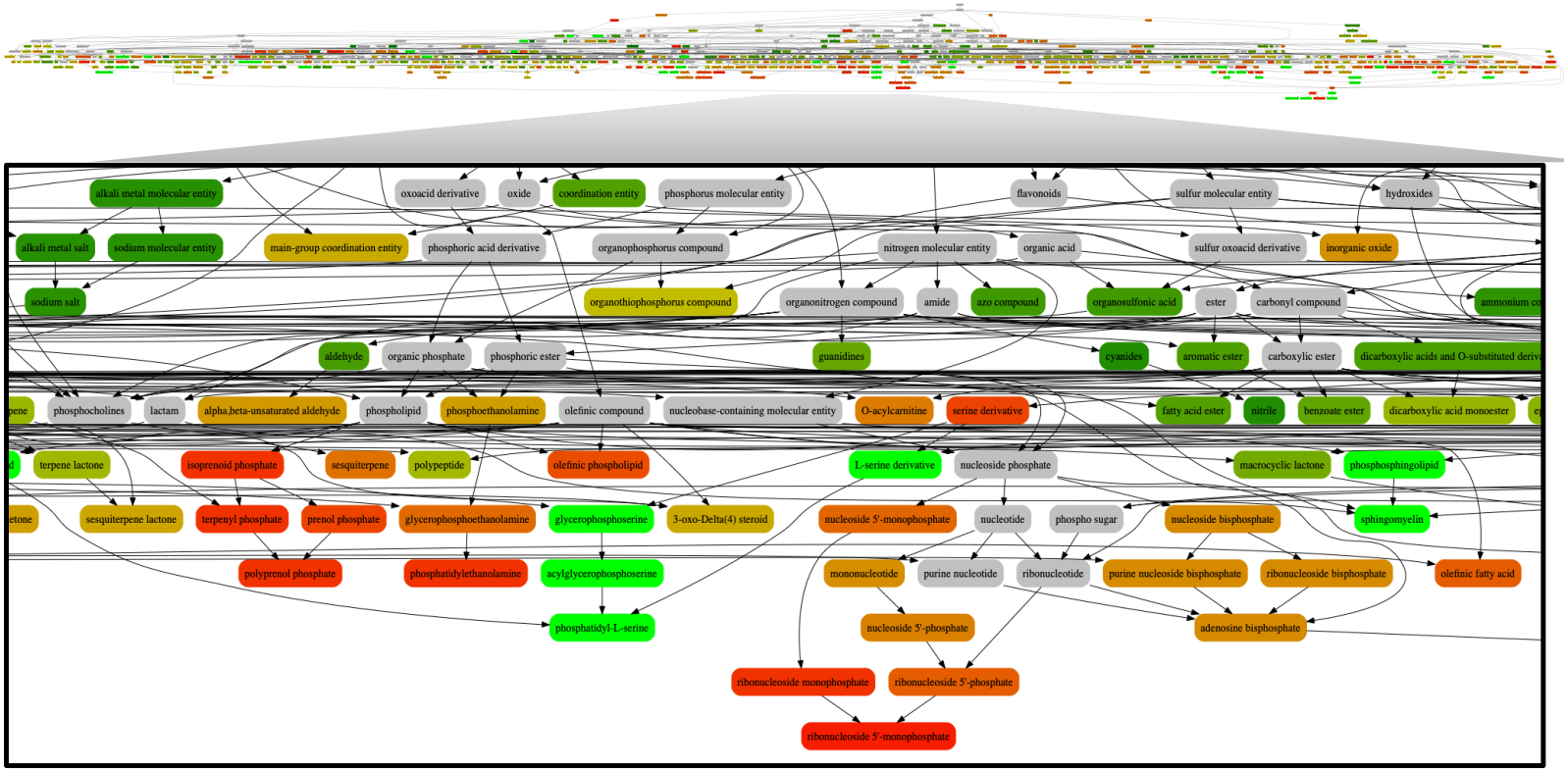


Figure 4.21: Zooming in on a subset of the predicted classes, colored by how many molecules are assigned to that class with red = fewer and green = more. Dark red = fewer than 10, Dark Green = more than 100, Orange = 10-100, Grey = not included in the 500 predicted classes but included in the visualization for hierarchical completeness.

haps larger targeted training sets would enable the network to better distinguish the features associated with these classes.

- Classifications into classes from both the inorganic and organic branches of the ontology, e.g. inorganic ion and steroid ester, or inorganic ion and dibenzopyrans.

As a final step, we then removed those 5 molecules that had conflicting classifications from the automated ontology extension, resulting in a consistent ontology extension. The extended ontology was manually inspected for correctness and no further problems were detected.

4.4.2.3 Evaluation on Datasets with Threshold

The above results indicate that the Electra model can be a good classifier for the 500 classes that were covered by the $\text{ChEBI}^{500*}_{v200}$ and $\text{ChEBI}^{500}_{v200}$ datasets. It does, however not cover all viable classes in ChEBI. As we have seen in Figure 4.8a, there are 854 classes in ChEBI that have at least 100 subclasses with SMILES annotations. We extended the scope of our tool by creating the $\text{ChEBI}^{854}_{v200}$ dataset that contains all classes in version 200 of ChEBI as labels and all their subclasses with annotated SMILES strings as instances. In order to extend the scope even further, we lowered this threshold to 50, which resulted in the $\text{ChEBI}^{1332}_{v200}$ dataset with 1332 classes.

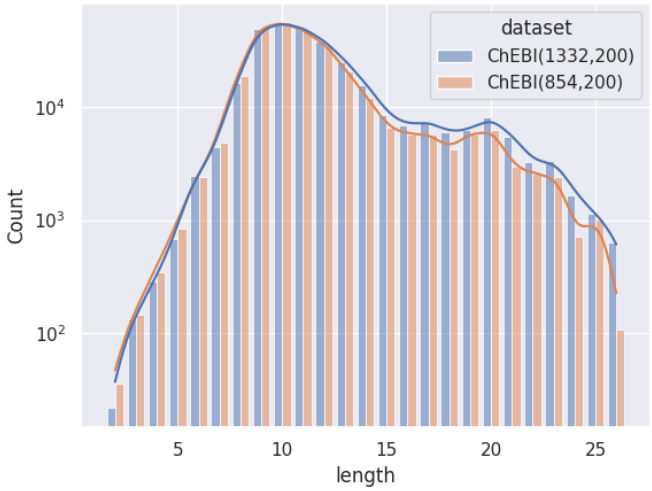


Figure 4.22: Distribution of path lengths (log scale) amongst non-redundant superclasses among molecular entities in the test set

Figure 4.22 illustrates the difference between $\text{ChEBI}^{854}_{v200}$ and $\text{ChEBI}^{1332}_{v200}$ by comparing the distribution of path lengths within the test set. These paths have been calculated based on the subsumption graph from each direct parent in the dataset, i.e., those that cannot be inferred from any other superclasses. It can be seen that when passing from $\text{ChEBI}^{854}_{v200}$ to $\text{ChEBI}^{1332}_{v200}$, the focus of the dataset shifts towards classes deeper down in the hierarchy. The way paths are counted is illustrated in Figure 4.23.

Preliminary tests indicated that the increased number of labels had a negative impact on the prediction quality for smaller classes. Therefore, we introduced an additional weighting scheme proposed by Cui et al. [123] that penalizes prediction error based on the number of members in each class C. This weighting mechanism assigns each class C of size $|C|$ an *effective number* $E_C = \frac{1-\beta^{|C|}}{1-\beta}$, which is an estimate of the space that a set of $|C|$ possibly overlapping volumes of size 1 would occupy. The intuition behind the metric is that new instances to already large classes have a smaller impact on the overall size than it would have for a smaller class. As proposed by Cui et al. we used the inverse of this *effective number* as a weighting with $\beta = 0.99$:

$$w_C = \frac{1-\beta}{1-\beta^{|C|}}$$

	Micro		**Macro**	
	unweighted	**weighted**	**unweighted**	**weighted**
$\textbf{ChEBI}^{854}_{v200}$	**0.9032**	0.8901	0.6070	**0.6372**
$\textbf{ChEBI}^{1332}_{v200}$	**0.9020**	0.9010	0.6022	**0.6552**

Table 4.7: Comparison of F1 scores on both datasets with different aggregation methods. The best result for each combination of dataset and aggregation method is highlighted in bold.

Two different metrics were used in the evaluation to comprehensively assess the performance of all models. The details of these evaluation metrics are explained in section 2.4.10. When comparing F1 scores (Table 4.7), both weighted and unweighted labels were considered for both datasets. This comparative analysis provided insight into how well the models handle the heterogeneous distribution of classes on both datasets. Micro F1 scores, which indicate overall precision and recall, showed similarities in both datasets. This metric provides a measure of how well the model performs given an individual chemical entity. The macro F1 scores, while lower than the micro F1 scores, provide insight into the model's performance on the overall class level. The lower macro F1 scores can be attributed to the inherent tendency

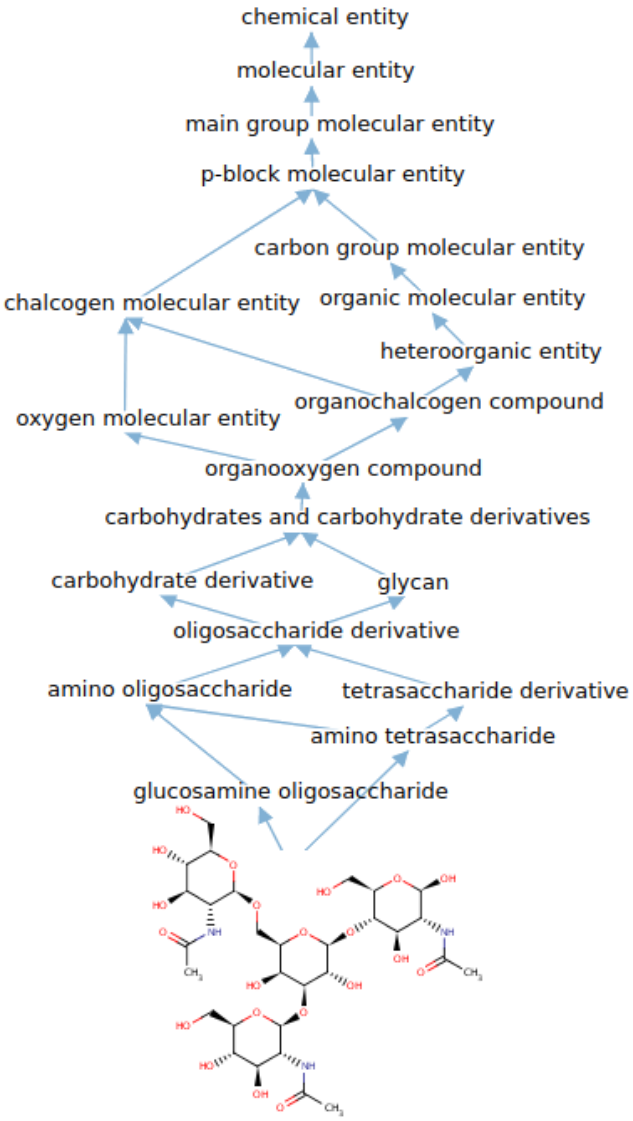
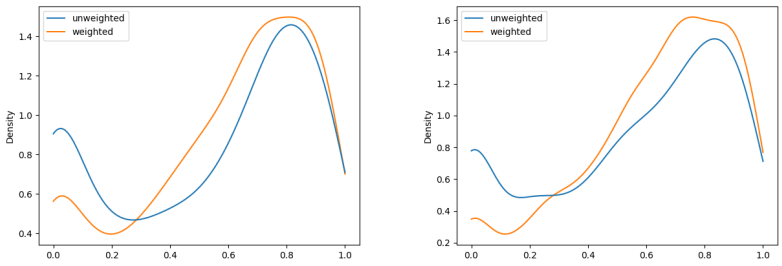


Figure 4.23: Example of subsumption paths from the molecular structure at the bottom (CHEBI:71342) to *chemical entity* (CHEBI:24431). CHEBI:71342 has two direct parents, namely *glucosamine oligosaccharide* (CHEBI:22485) and *amino tetrasaccharide* (CHEBI:59412). There are nine paths of length 12 and nine paths of length 14 from these parents to *chemical entity* (CHEBI:24431).



(a) ChEBI$^{854}_{v200}$ (b) ChEBI$^{1332}_{v200}$

Figure 4.24: Distribution of F1 scores among classes on weighted and unweighted on ChEBI$^{854}_{v200}$ and ChEBI$^{1332}_{v200}$.

of the model to focus more on predominant classes, a challenge that mirrors the broader problem of bias in machine learning more generally.

We aimed to mitigate this tendency by introducing additional weighting, which would allow for a more balanced assessment, especially for datasets with unbalanced class distributions. The evaluation metrics showed an increase in the performance regarding the macro F1 evaluations when employing additional weighting. This improvement was more pronounced for ChEBI$^{1332}_{v200}$ compared to ChEBI$^{854}_{v200}$, with the former benefiting more due to its hierarchical structure with a larger number of classes further down the hierarchy. Figure 4.24a and Figure 4.24b show the density plot of the distributions of F1 scores for both datasets with and without weighting. It can be seen that the weighting improved predictions in both cases and lowered the number of cases in which no correct predictions were made.

Evaluating the specificity of model predictions

Datasets that are produced with a lower threshold will always cover more classes than ones with a higher threshold but the complex structure of ChEBI hinders an easy intuition of the actual structure of the resulting ontology fragment that is covered by the prediction model. In order to quantify the depth that was added by this lowering of the threshold from ChEBI$^{854}_{v200}$ to ChEBI$^{1332}_{v200}$, we analyzed the path length of classes along the class hierarchy to the root. For each of the molecules in the test set of ChEBI$^{1332}_{v200}$ and ChEBI$^{854}_{v200}$, we evaluated the path lengths from the directly asserted parent in the ontology to the predicted parents in the calculated model predictions. The best case is that the distance is 0, i.e., that the predicted parent class is the same as the asserted parent class. Figure 4.25 shows the distribution of path lengths between classification parents and predicted parents in the two datasets.

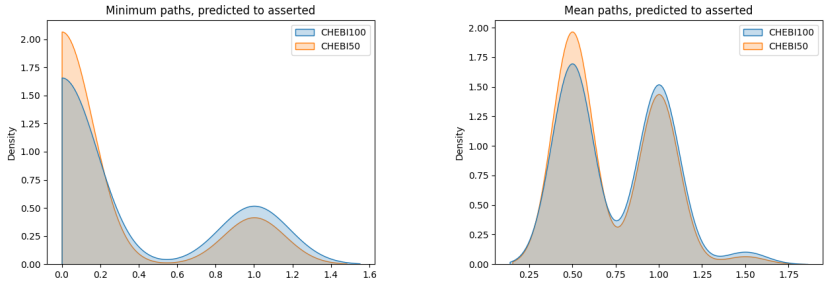


Figure 4.25: Minimum (left) and mean (right) path lengths between the asserted parent for a SMILES string and the predicted parent in the two datasets.

For a given molecule and a correct prediction C_1 by the ChEBI$^{1332}_{v200}$ model, all correct predictions of the ChEBI$^{854}_{v200}$ model for the same molecule are considered. If the ChEBI$^{854}_{v200}$ model predicts a subclass C_2 of C_1, then its prediction is considered more specific. If the ChEBI$^{854}_{v200}$ model predicts a superclass C_2 of C_1, then its prediction is considered less specific. In case both models agree on their predictions, their predictions are equally specific. In the last case, the ChEBI$^{854}_{v200}$ model makes no prediction that is in the same taxonomic branch as C_1. More specifically, the ChEBI$^{854}_{v200}$ model does not correctly predict some class C_2 such that C_2 is comparable to C_1 with respect to the partial order provided by the taxonomic structure of ChEBI. In this case, the predictions of the two models are considered as not comparable. As Table 4.8 shows, the predictions of the ChEBI$^{1332}_{v200}$ model are more specific in 21.69% and less specific in 8.43% of the cases compared to the ChEBI$^{854}_{v200}$ model. In most cases, however, there is no difference between the models.

	more specific	**less specific**	**equally specific**	**not comparable**
Amount	5,021	1,951	14,448	1,724
Percentage	(21.69%)	(8.43%)	(62.43 %)	(7.45%)

Table 4.8: Analysis of specificity between predictions made by the systems trained on ChEBI$^{1332}_{v200}$and compared with ChEBI$^{854}_{v200}$.

The three different approaches to evaluate specificity lead to a consistent result: in comparison to its predecessor the model trained on ChEBI$^{1332}_{v200}$ provides more specific predictions. However, its increase of coverage of ChEBI by 56% does not translate into an equally large increase in the specificity of its predictions. Unfortunately, although we did observe an increase in the specificity of model predictions across a range of metrics, the observed increase in prediction specificity was only moderate. This result can largely be attributed to two main causes.

First, $\text{ChEBI}^{1332}_{v200}$ extends $\text{ChEBI}^{854}_{v200}$ with classes that have between 50 and 99 examples (i.e., classes that have between 50 and 99 members, that is subclasses that are annotated with SMILES in ChEBI). $\text{ChEBI}^{1332}_{v200}$ is trained on 129,187 molecules. As a result, only between 0.039% and 0.077% of molecules in the dataset are members of any given class amongst those that are in $\text{ChEBI}^{1332}_{v200}$ but not in $\text{ChEBI}^{854}_{v200}$. Hence, any one of these additional classes is irrelevant for the majority of classification tasks. In fact, only 18.79% of all molecules in the dataset are a member of any of the 478 additional classes. Thus, extending $\text{ChEBI}^{854}_{v200}$ by 478 of these relatively rare classes can only be expected to have a limited impact on the overall performance. Second, since our model is trained based on datasets generated from ChEBI's taxonomy, it learns to imitate ChEBI. And because ChEBI contains a lot of examples of chemical entities that are classified on a quite high level, our model imitates that lack of specificity. For example, *lactam* (CHEBI:24995) has 7,328 direct subclasses. Thus, while ChEBI provides a taxonomy that enables the classification of different types of lactam, given the vast number of molecules that are directly classified as lactam by ChEBI, it is difficult for a machine learning model to learn the subclasses of lactam. Lactam is not unique, e.g., azamacrocycle (CHEBI:52898, 7,246 direct subclasses) and peptide (CHEBI:16670, 6,199 direct subclasses) are two other examples of classes that are used to classify molecules on a high level of generality. Note that these general classifications are usually not the result of ChEBI's manual curation process, but the result of semi-automated processes that are already adopted in the development process, largely using the ClassyFire tool. A specific mitigation strategy for this problem in the future might be to exclude these overly general classes from the training data and subsequently re-classify all the children of the classes in a targeted ontology specificity improvement step. An alternative strategy in the future to further increase the specificity of predictions would be to include classes with even fewer examples (e.g., classes with at least 25 subclasses that are annotated with a SMILES), although there are additional challenges to training models to learn from such a small number of examples. In general, as the development of ChEBI progresses and ChEBI is improved by providing more examples of more specific classifications, the ability of our model to provide more specific classifications will also in turn improve.

4.4.2.4 Comparison between ChEBI versions

The purpose of our tool is to provide a system that is more adaptable to newer versions of ChEBI than the currently used ClassyFire tool. Our results indicate that the model can be scaled to larger portions of ChEBI. However, the role

of the ontology version has not yet been investigated. Therefore, we repeated the data extraction process for $\text{ChEBI}^{854}_{v200}$ on version 148 of ChEBI.

	No. classes	**Training set size**	**Micro F1 (all cl.)**	**Macro F1 (all cl.)**	**Macro F1 (common)**
v148	709	80,639	0.8546	0.5133	0.5200
v200	854	129,187	**0.9032**	**0.6070**	**0.6231**

Table 4.9: Comparison of number of classes, training size and F1 scores with different aggregation methods for $\text{ChEBI}^{709}_{v148}$ and $\text{ChEBI}^{854}_{v200}$. The macro F1 score was calculated for all classes predicted by the model and, for comparison, also on the subset of 701 classes that appear in both ChEBI versions. The best result for each score is highlighted in bold.

Table 4.9 outlines the dataset sizes and classification performance for both versions. Following four years of development, more classes in $\text{ChEBI}^{854}_{v200}$ now surpass the threshold of having at least 100 subclasses associated with SMILES compared to $\text{ChEBI}^{709}_{v148}$. This has resulted in an additional 145 classes in $\text{ChEBI}^{854}_{v200}$ compared to $\text{ChEBI}^{709}_{v148}$. Notably, ChEBI version 200 provides over 50% more training data, contributing to an overall higher F1 score with both micro- and macro-aggregation.

However, the comparison of macro F1 scores between $\text{ChEBI}^{709}_{v148}$ and $\text{ChEBI}^{854}_{v200}$ is somewhat constrained due to the inclusion of different classes in each version. For instance, the improvement in the macro F1 score from version 148 to version 200 may be influenced by the latter version containing classes that are inherently easier to learn. To facilitate a more meaningful comparison, we also computed macro F1 scores specifically for the 701 classes shared between $\text{ChEBI}^{709}_{v148}$ and $\text{ChEBI}^{854}_{v200}$. With respect to these shared classes, the macro F1 score is 10% higher for the model trained on version 200.

These results are important because they highlight two important aspects of our model in relation to ontology development. For one, the results indicate that the model benefits from a larger amount of training data. But it can also be seen that, while the increase in classes between $\text{ChEBI}^{854}_{v200}$ and $\text{ChEBI}^{1332}_{v200}$ had little effect on the predictive performance, the same was not true between $\text{ChEBI}^{709}_{v148}$ and $\text{ChEBI}^{854}_{v200}$. In the latter case, the performance of the model improved significantly. The high quality of the manual ontology development did therefore fix important issues that were detrimental to the prediction quality of our model. This further highlights the importance of ongoing ontology development that we aim to support with our proposed system.

4.5 The ChEB-AI Tool

My contributions

I am the founding developer and current main developer of the ChEB-AI library. Whilst, in the meantime, there is a larger team of developers, all data processing methods and models that I developed in the context of this thesis are part of this tool and were implemented by me. The implementation of the initial ChEB-AI interface as well as the transition to PyTorch Lightning were planned and executed by me.

The heterogeneous nature of data sets and models in machine learning often presents an unnecessarily large barrier to entry into this domain. A significant amount of implementation work is often required to incorporate new data into existing models. Libraries such as PyTorch [124] and tensorflow [125], which provide a standard interface for data and models, have already taken an important step towards making these methods easier to use. In the domain of ontology-based learning, this poses a particular challenge. Here it is often necessary to use methods that go beyond conventional data processing. As we have already discussed in Section 4.1.2, for example, it is necessary to calculate the transitive closure of subsumption relation within the ChEBI ontology in order to assign the correct labels to all molecules. We therefore consolidated the data processing, training and evaluation scripts that have been used in the experiments shown in this thesis into a single, open-source Python library: The ChEB-AI framework[5].

The first version of this library included a custom interface for data and model management as well as a training setup that was based on PyTorch version 1. In the course of the upgrade to PyTorch 2, we decided to switch to the existing Pytorch Lightning Framework [126], because it allows frictionless training on multiple GPUs and cluster nodes. The existing modules of ChEB-AI were then rewritten so that they corresponded to the new Lightning interface. As usual in the Lightning Framework, the ChEB-AI training interface consists of three core components: A model, a data loader and a trainer. The model is one of the core components and, as the name indicates, defines the machine learning model used in a training task. Its functionalities, however, go beyond that. A Lightning model also handles the computation of loss as well as the configuration of optimizers used during training, such as Adam. We implemented a based class *ChebaiBaseNet* that extends the the basic *LightningModule* to automatically log important metrics during training such as the F1 score with micro and macro aggregation.

[5] `https://github.com/ChEB-AI/python-chebai`

The data loader in Lightning enables the reproducible generation of datasets, with automated downloads of missing files and splits into test, train and validation sets. We implemented custom data loaders for all multi-label datasets presented in this chapter. The framework also allows the frictionless integration of other ontologies which enables the usage of this framework beyond the methods described in this thesis. In order to incorporate a wide variety of datasets and data processing mechanisms, ChEB-AI introduces the notion of "data readers". A reader defines the way data is read from a data source. The different tokenizers discussed in Section 4.2 are examples for different readers. This mechanism allows the comparison of the influence of different data encodings similar to the one between the BPE tokenizer and Chem tokenizer reported in Section 4.4.2. Our custom dataloaders also enable different reasoning capabilities, such as the calculation of the transitive closure of an ontology's subsumption hierarchy. Contrary to mOWL [127], a comparable tool for ontologies in the domain of bio-medicine, we do not incorporate the OWL API into this process. This decision was made, because our experiments have indicated that the OWL API was not performant enough to handle ontologies of ChEBIs size. Instead, we use the *fastobo* library [128], to parse the OBO-representation of the ontology and implement important reasoning steps as part of the ChEB-AI library. We are, however, currently investigating the integration of Horned-OWL [129]. Preliminary tests have indicated that this tool vastly outperforms the OWL API on large ontologies and, as of recently, there is also a python-based interface[6], allowing for more frictionless communication with the tool. A trainer ties the data loader and the model together and handles the communication between both. Most importantly for us, the trainer also defines the training strategy, which controls how multiple GPUs and cluster nodes can be used in parallel during training. Lightning's trainers are mostly used as-is in ChEB-AI. We do, however, define a default trainer configuration that enables the logging and training configurations used by us.

We hope that this infrastructure will allow a frictionless integration of datasets, ontologies and models from different domains. The generic data interface allows for an easy integration of new methods into this framework. In our current line of research, we also aim to integrate more formal reasoning into the system. This, naturally, involves reasoning with the ontology as mentioned before. However, we are also exploring formal descriptions of chemical entities and are, therefore, also implementing bindings to the *Heterogenous Tool Set* (HETS, [130]) via our in-house proof library, *gavel* [66], in order to support more expressive logics.

[6] `https://jannahastings.github.io/py-horned-owl/`

4.6 Chebifier

My contributions

The work in this section is based on collaborative work [115]. My contributions were the conceptualization training, implementation of the models already presented, as well as the conceptualization, design, and implementation of the Chebifier web interface – including both frontend and backend. This does not include the design, execution or evaluation of the user study. Furthermore, I was the first author of the related journal article [115].

In the previous sections, we presented different approaches towards automated ontology extension that are fine-tuned to represent the consensus within a specific ontology. These models, however, require some knowledge on how to install and interact with the python-based ChEB-AI framework. The purpose of this tool is to lessen the workload that ontology development imposes on domain experts. An expectation for them to acquire these skills is therefore counterproductive. Therefore, we implemented an easy-to-use interface that allows domain experts to simply input the SMILES representation of a molecular structure and receive the predicted subsumptions as predicted by the model: Chebifier.

4.6.1 The Chebifier System

The primary use case that Chebifier is intended to address is as follows: An expert from the domain of chemistry or biology notes that a chemical that is important to their line of work is not represented in the ChEBI ontology. They know the precise molecular structure, but not the correct classification according to ChEBI. This user can represent this chemical structure using the SMILES notation discussed in Section 2.3.1, visit the Chebifier website[7] and retrieve a classification in ChEBI.

Figure 4.26 shows the system architecture of Chebifier. The Chebifier user interface is a web application developed based on a simple Python Flask server with a JavaScript React front-end. There are two ways in which users can enter the SMILES strings. Firstly, they can be entered directly into a ready-made table form. Chebifier also allows a large number of chemicals to be classified at once. Accordingly, users can also upload a text file containing one SMILES string per line. These are then automatically transferred to individual lines in the table.

[7] `https://chebifier.hastingslab.org/`

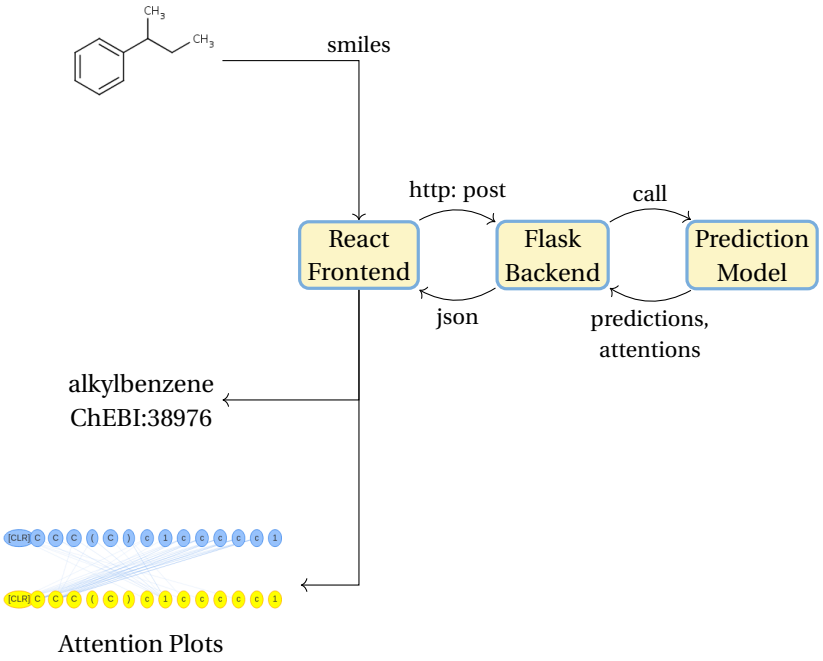


Figure 4.26: System architecture of Chebifier

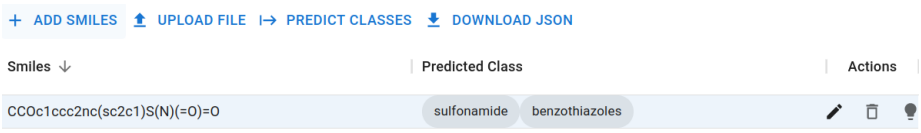


Figure 4.27: The user interface of Chebifier with example input.

By clicking on the "Predict Classes" button, these SMILES strings are transferred to the backend with an HTTP post request. Here, the corresponding connected model is called in order to obtain the corresponding classification predictions.

The Chebifier system is based on a prediction model that was developed with the help of the ChEB-AI library. The connection was implemented in such a way that any trained ChEB-AI model can be used for predictions with minimal effort. Chebifier 1.0 was based on a model trained on $\text{ChEBI}^{854}_{v200}$ using the Electra model presented earlier. As we will elaborate later, feedback from our user study indicated a desire to include more specific classes from ChEBI. In response, we created the $\text{ChEBI}^{1332}_{v200}$ dataset, which includes all classes in ChEBI (release 200) that have a minimum of 50 SMILES-annotated subclasses. The reduction in the size of members required allowed the dataset to expand to a wider range of classes, resulting in 1,332 class labels that the model is able to predict. Chebifier 1.1 uses a model that is trained on $\text{ChEBI}^{1332}_{v200}$ with weighting.

The calculated predictions are then returned and displayed as bullets next to the input form. These bullets also act as a link to the corresponding ChEBI classes, allowing experts to manually check the predictions directly if required. Figure 4.27 shows the interface after this workflow. The user submitted a string that represents the class of *ethoxzolamide* (CHEBI:101096). The system predicted the superclasses *sulfonamide* and *benzothiazoles*, which matches the superclasses according to the ChEBI ontology.

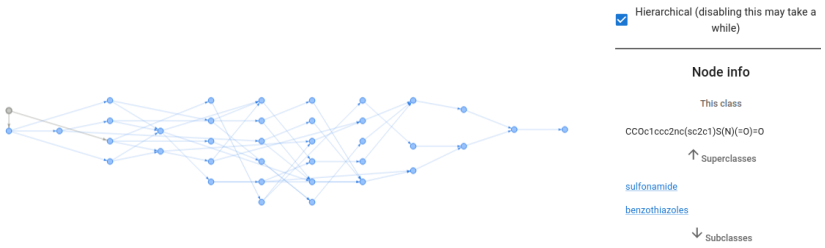


Figure 4.28: The ontology fragment for *ethoxzolamide* as predicted in Figure 4.27. The interactive visualization displays class labels when the pointer hovers over a node.

Chebifier can be used to provide classification suggestions to improve the speed of the manual curation process of ChEBI. After classification, Chebifier also displays an extended version of the ChEBI ontology. As can be seen in Figure 4.28, this is represented as a directed graph in which existing classes are represented by blue nodes and the new classes by grey nodes. This graph representation is also interactive so that users can arrange the nodes themselves for better readability if this is not sufficiently good despite the automated lay-

out. Further information on the classes can be obtained by clicking on the corresponding node. These are then displayed in a sidebar.

To make the system easier to understand, detailed information is also provided for each classification in a molecule-specific view. This detail view, parts of which are shown in Figure 4.29, can be pulled up by clicking on the light bulb icon next to the classification bullets. The molecular graph associated with the molecule, as rendered by RDKit, is displayed here. There is also an extended version of ChEBI, which has only been extended by the class for which the details are displayed. Additional information about the classifier is also displayed for further interpretability. Since a transformer-based model was connected in the previous versions, the attention weights are displayed as a network graph. We are planning to develop further explainable systems in the future, for which the corresponding classification explanations can be displayed here.

Since Chebifier is itself trained from the content of ChEBI, future extensions of ChEBI will improve the performance of Chebifier. Thus, in the long run, our approach has the potential to create a virtuous feedback cycle of ontology extension, checking and then improvement of the classification model, which serves to lay the foundations for semantic chemical discovery research.

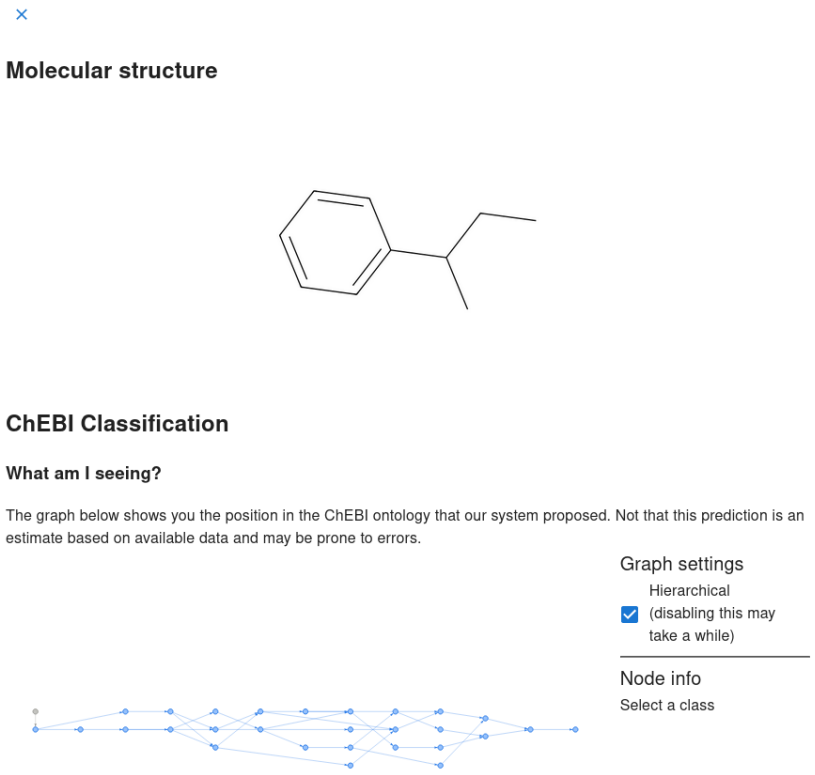
4.6.2 User study

An anonymous online user survey in the English Language was conducted from June to August 2023 using Qualtrics[8]. Following a purposive recruitment strategy, we engaged participants via LinkedIn, Twitter, and direct emails. The recruitment strategy included invitations to the curators of ChEBI, the PubChem team, the developers of key partner databases such as Rhea [131], and members of the chemical informatics community who had previously shown an interest in chemical classification.

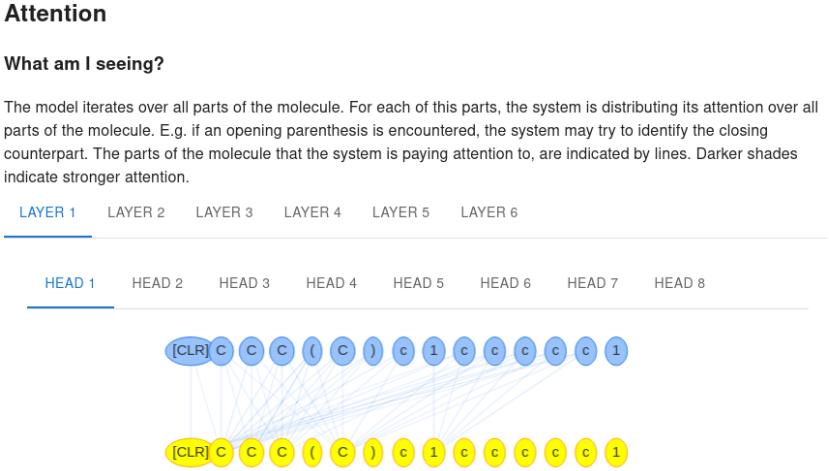
The objectives of the user survey were threefold. First, we aimed to assess user satisfaction with the accuracy and exactness of the automated classifications. Second, we aimed to gather insights on the user interface, including associated visualizations. Third, we aimed to explore the acceptability and reception of a machine learning-based application for the problem of predicting structure-based chemical ontology classifications.

The survey questionnaire was divided into two parts, collectively comprising a total of six questions with open-text response options. It was estimated that respondents would require approximately 10 minutes to complete the survey. The initial part of the survey directed participants to engage with the Cheb-

[8] www.qualtrics.com



(a) Molecular graph and extended ontology



(b) Attention network for first attention head in the first layer

Figure 4.29: Screenshots of Chebifier's detail view for predictions of *sec-butylbenzene* (CHEBI:35097)

ifier tool and add one or more SMILES strings representing molecular entities of their choice, and then execute class prediction. Once the prediction was returned, respondents were asked to rate its correctness and appropriateness. Subsequent to the prediction step, participants were directed to open the individual details page for a single prediction. On this page, a visualization of the molecular structure of the input SMILES was displayed along with a subset of ChEBI, relevant to the molecule's direct parents. Finally, the prediction model's internal attention parameters were displayed. Participants were asked to evaluate the utility of the visualization representing the subset of the ChEBI hierarchy. Furthermore, they were encouraged to explore the attention clusters, encompassing various heads and layers, and asked for recommendations for improvement. Concluding the survey, respondents were offered an opportunity to provide further suggestions to enhance the Chebifier tool as a whole.

4.6.3 Results

A total of 12 participants engaged in the survey, testing a diverse range of SMILES inputs and offering useful suggestions on both the user interface and tool functionality. Out of these, 9 participants (75%) rated the classifications as either correct or partially correct as well as appropriate for the provided input molecules. In contrast, one solitary user (8%) noted inaccuracy, while two participants (17%) refrained from specifying their responses. There were instances of missing or inaccurately assigned classifications:

- *"Yes, they are true but do not describe the most important point. This is a transition metal complex! Hence, transition metal should come as classification as well."*
- *"Yes, but only one classification was made and many others are equally true, for example D-galactose, cerebroside, β-galactosylceramide."*
- *"Four answer provided. Three are correct. One is not the organocarbonyl was not correct. I suspect it called the amide a as an organocarbonyl."*

Considering the correct classifications, 7 participants (58%) acknowledged the classification's chemical significance, either fully or to a partial extent. Conversely, 2 users (17%) expressed no utility in the classification, while 3 users (25%) did not further elaborate their response. However, a recurring theme was that the classifications were overly generalized, highlighting the need for a finer level of granularity:

- *"Yes they are meaningful if someone want to look at the general classes but if users want to go into a more detail level of granularity, especially for natural product classes, then ClassyFire may be a better option."*
- *"I see no reason why the one classification that was made is more meaningful than the other possible classifications."*
- *"I tested several lipids and most of the time the class predicted was a close parent. However, there were cases in which the class predicted by the tool was true but too general [...]."*

Regarding the efficacy of visualizing a subset of the ChEBI hierarchy, merely 2 users (17%) perceived it as valuable. In contrast, 5 users (42%) abstained from answering, while the other 5 users (42%) found the visualization unsuitable for their purposes. Their feedback highlighted the superiority of other established software and suggested improvements such as a vertical orientation display and the inclusion of labels to enhance the interface design:

- *"It is useful. I would recommend to turn the display of the hierarchy into a vertical tree marking the root class in another color such as red, and the molecule been classified in a color that the user can clearly see where is the instance/molecule located in the hierarchy. To read the labels of the two first parent layers would also increase clarity, so I would make them static and readable in the graph."*
- *"Not really [useful] - none of the names can be seen without mousing over, so it cannot be scanned and taken in. I also prefer vertical orientation.*
- *"Only first nodes, after node organic molecular entity I think it is no longer useful. I guess it depends on what each user is looking for."*

Moreover, for the attention clusters, 3 users (25%) detected disparities in attention allocation among different attention heads, with one conjecturing that this variance might stem from layer alterations:

- *"This seems to vary by altering layers, and HEAD assignments, so I'd suggest seeing the aspirin SMILES prediction by yourselves."*

Conversely, the remaining 9 users (75%) either deemed the function unhelpful or refrained from assessing it due to a lack of expertise:

- *"I don't understand how to interpret this visualization. What is a head?"*
- *"I do not understand this well enough to answer."*

Collectively, suggestions for enhancing the tool encompassed several areas, including the need for improved accuracy and improvements to the interface design:

- *"I think it would be easier if a user adds the SMILES and the tool automatically predicts the class instead of the user clicking on the save and predict class buttons first."*
- *"Putting in a large group of structures and clustering them in meaningful ways would be useful for us."*
- *"Thanks for working on this much needed tool. I would suggest to adjust it for closer parents, the higher levels can be drawn from the hierarchy."*

As a direct response to the consistent user feedback that the classifications provided by Chebifier were often correct, but too general, we developed Chebifier 1.1, which – compared to version 1.0 – increases the coverage of ChEBI by 56% without significant changes in accuracy. Since the additional classes are all lower in the taxonomic hierarchy of ChEBI than previously covered classes, Chebifier 1.1 is able to provide more specific classifications than the earlier version and thus addresses a major concern raised by our users. We further illustrate the improvement in the specificity of the predictions in the next subsection.

4.7 Discussion

Semantic resources such as ontologies are key for assigning meaning and interpretation to data, which in turn drives research. Yet the maintenance of these knowledge resources is a challenging task that requires significant contributions from domain experts, making it slow and labor-intensive. The pace of scientific discovery, however, is continuously accelerating, driven in part by the growing integration of artificial intelligence in all research areas [132]. Supporting the evolution of ontologies at sufficient pace and scale requires semi-automated tools and approaches that are faithful to the existing domain knowledge and ways of thinking about scientific content, while also having the potential to learn directly from data. In this chapter, we presented a novel method that allows the training of a system based solely on a given ontology with structural annotations. This approach offers a strategy to navigate this tension in a sustainable 'AI in the loop' fashion, with the potential to be kept up to date automatically and significantly accelerate the maintenance of ontology-based knowledge resources in chemistry to keep up with the pace of discovery.

ChEBI already makes use of an automated tool to extend its coverage beyond the manually curated core, namely ClassyFire, a rules-based system in the form of - among others - manually defined SMARTS strings. This limits the extensibility of ClassyFire as an addition of new classes requires the definition of new SMARTS strings or even more complex rules. Our analysis of the model performance on $\text{ChEBI}^{709}_{v148}$ and $\text{ChEBI}^{854}_{v200}$ showed a newer version of the ontology with a larger number of classes is not just possible, but may even improve the model performance. The deep-learning-based approach that we presented can therefore overcome the limitations of rules-based approaches by allowing a dynamic creation of classifiers based on a given existing ontology structure. However, the dataset that could be used for fine-tuning is still just a subset of the ontology corresponding to up to 1,332 classes, thus falling short of the full scope of the entire ontology. Further reducing the selection threshold, however, is likely to diminish the predictive power of the system because the model may have fewer examples for classes that have very unusual or paradigmatic structural features. This could be seen as a limitation of the applicability of this method, as many of the classes that chemists may be interested in may have fewer members. A manual review of ChEBI has shown that there are some classes with a surprisingly high number of direct subclasses. The class of *peptide* (CHEBI:16670) has more than 1,000 direct subclasses, a majority of which have been added by third parties, possibly using automated methods such as ClassyFire. Targeted curation efforts could be used to revise this structure of the ontology and add to important classes for any given use case which allow our approach to cover a deeper, more relevant fragment of this domain.

Our overall analysis has shown that the Chebifier tool presented in this work can be a useful integration into the ontology development process. Our analysis in this work was limited to the ChEBI ontology. However, the only requirement that an ontology has to fulfill is the presence of structural annotations that contain information about the intended semantics of the annotated class. As is the case for ChEBI, not all classes need to be annotated in that way. The Protein Ontology[9] (PRO) can be used with a very similar architecture that uses a peptide-based tokenizer instead of an atom-based one. It is also worth investigating to which extent this approach can be extended to the Gene Ontology, considering the promising results around DeepGO [98], DeepGoZero [101] etc. Conversely, the methods applied to the Gene Ontology may also be applicable to ChEBI. We are also investigating the possibility of including image data for examples of members of a class. Specifically, we aim to use an ontology from the domain of ornithology that is linked with visual data of birds and feathers to train a system that enables image-based ontology classification.

[9] https://proconsortium.org/

5
Semantic Support

In the previous chapter, we explored the ways in which machine learning can support ontology development. Machine learning approaches have led to significant developments in many areas of science. However, it is important to emphasize that machine learning is not a "golden hammer" that can simply be applied to any existing problem. In order to successfully apply such a system, appropriate prerequisites must be met. For more modern approaches, this often means that large amounts of high quality data must be available. However, this is not possible in all domains. Furthermore, even with the successful application of machine learning methods, it is only possible to a limited extent to gain insights into the fundamentals on the basis of which a model makes its predictions. Approaches that incorporate human knowledge alongside learning from data, which have been called hybrid, knowledge-aware or *informed* [133], have the potential to improve the correspondence between what the model learns and the structure of the human world, which in turn allows the model to learn more generalisable representations from smaller datasets.

In this chapter we want to investigate whether ontologies can be used to address some of these problems.

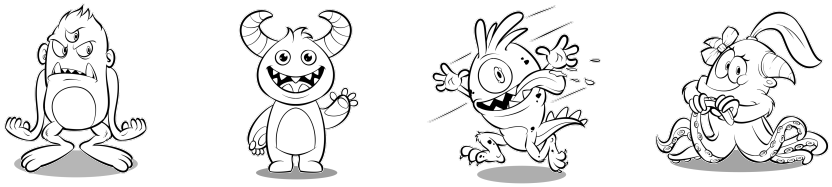
5.1 Ontology Pre-Training

My contributions

The following section summarizes the results of collaborative work published in [134]. My contributions to this work were the implementation, training and evaluation of all models.

Human knowledge is carefully curated into ontologies [135, 55], making them a prime candidate source of knowledge to incorporate into learning. This

naturally raises the question of whether this knowledge can also be actively used to support machine learning. When combining ontologies and machine learning approaches, however, obstacles are quickly encountered. An essential problem is the alignment between the ontology and the way in which the world is represented in the input data and the model. Ontologies are often based on important philosophical considerations, while data models are often optimized for efficiency, e.g. of read and write accesses. Both approaches obviously have their raison d'être in their specific domain.



yes no yes no

Figure 5.1: A fictional dataset for Woblox-Classification. The labels below each monster denote whether it is classified as a Woblox or not.

However, we argue that it may be beneficial, but not absolutely necessary to combine these two views. Imagine the following binary classification problem: A system is to learn to recognize an imaginary type of monster based on images: The Woblox. As training data for this classification problem, the data set shown in Figure 5.1 is given. As in real learning tasks, we have a variety of features that can potentially be important for this classification, e.g. the number of eyes or the presence of teeth, horns or limbs. However, the small amount of does not data provide the necessary basis for statistical methods like the ones discussed in Section 2.4.

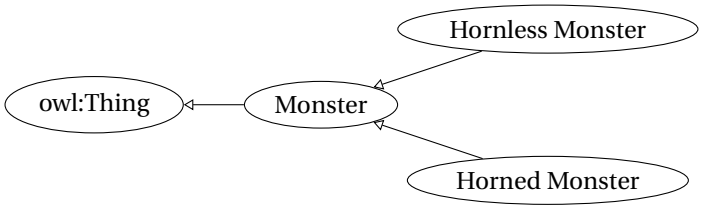


Figure 5.2: A fictional ontology for the domain of monsters

But now let us also imagine that there is an ontology from the monster domain that has been developed by experts for many years, as shown in Figure 5.2, for example. Notably, the term "Woblox" does not occur in this ontology. We can, however, learn that one of the key features that experts use to classify monsters is the absence or presence of horns. We may therefore pay larger attention to those features when trying to classify our monsters and thereby use

the expert knowledge contained within the design decisions of the ontology despite the absence of our specific target labels from the ontology.

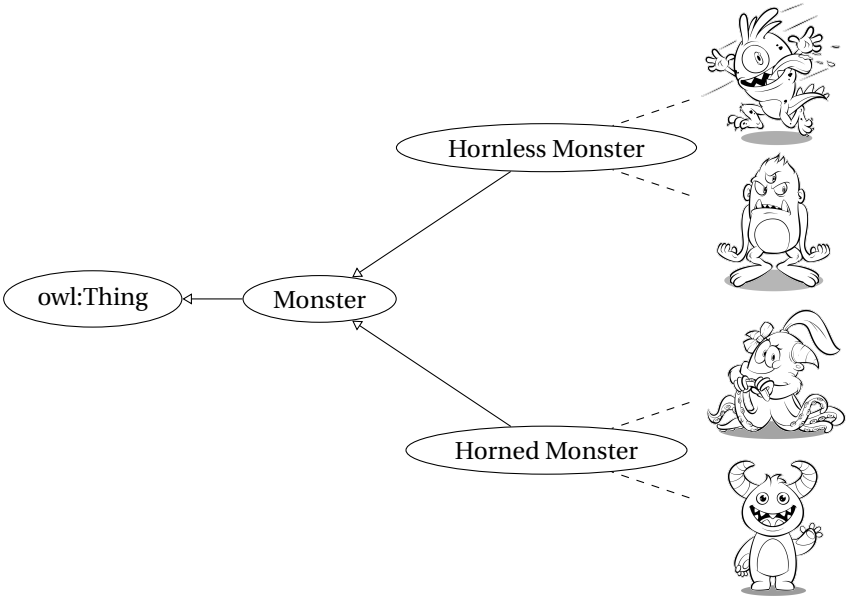


Figure 5.3: A fictional ontology for the domain of monsters with instances

But this leads to the next important question: How can a model use an ontology? Because simply "reading" the ontology is not enough. A human reader could certainly read the names of the classes and derive the intended semantics of a class, as essential terms such as horns are already known and can be linked to the corresponding features in the illustrations. A machine learning model cannot do this as easily because it cannot know the intended semantics. But as we discussed in Chapter 4, it is possible to teach a system the intended semantics of classes if we have an ontology that has structural annotations. Figure 5.3 illustrates the ontology from Figure 5.2 with added instances as data points. Using our approach for ontology extension would allow the model to learn the intended semantics of the classes *Hornless Monster* and *Horned Monster*.

In the following, we will show a way to apply this concept to the domain of chemistry. As before, we used the ChEBI ontology to train a model on the ontology extension described earlier. As in the previous chapter, we consider an ontology with structural annotations as depicted in Figure 5.4. In order to successfully solve this task, the model has to detect those structural features that were used to make the distinctions made by the ChEBI developers. As in the monster example above, the system therefore has the opportunity to learn about properties relevant to the domain without a specific task that requires

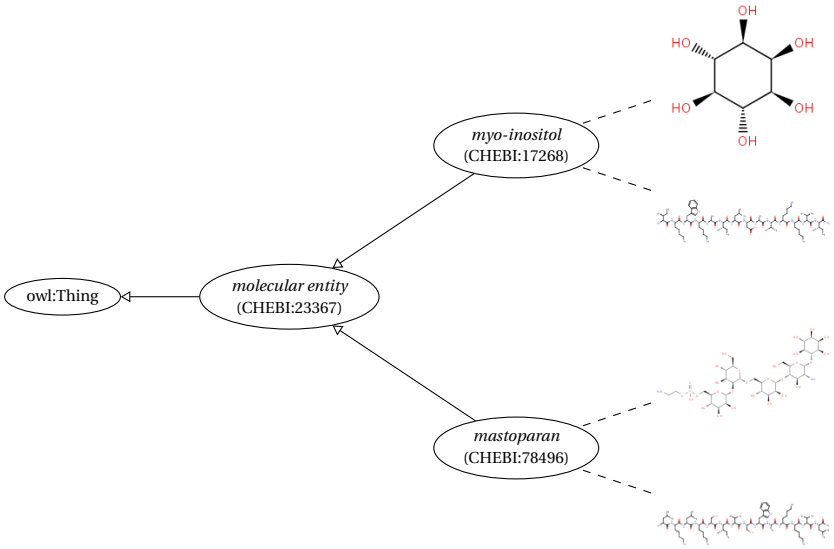


Figure 5.4: A fragment of ChEBI with four structural annotations. Intermediate classes between *molecular entity* (CHEBI:23367) and the two subclasses *mastoparan* (CHEBI:78496) and *myo-inositol* (CHEBI:17268) have been omitted.

a large, task-specific labeled data set. This approach is therefore versatile and can applied to a variety of different problems from the domain of chemistry. The next step is to check whether the model is then also able to successfully reuse this information when learning a more specific task. In order to do so, we evaluated our model on the task of toxicity prediction.

5.1.1 Datasets

We will use the Tox21 datasets as our benchmark dataset. It was created by the National Center for Advanced Translational Sciences (NCATS) of the National Institutes of Health (NIH) and constitutes a widely used benchmark for research on machine learning for toxicity prediction from small molecules [82]. Notably, there are currently two versions of the Tox21 dataset available in benchmarks. The first one originates from the Tox21 Challenge that was conducted in 2014. This dataset consists of three different subsets, one for training and two for different steps of the challenge evaluation. In our study, we will use the “testing dataset" that was used for an initial ranking of models as a validation set and the dataset that was used for the final evaluation as our test set. This version of the Tox21 dataset suffers from several issues regarding the consistency and quality of different entries [136]. A more modern version

of this dataset has been published as part of the MoleculeNet benchmark [77]. This version of Tox21 consists of only 7,831 molecules. We split this dataset into a training (85%), validation (7.5%) and test set (7.5%). There are, however, still two major challenges that need to be considered when working with this dataset. First, the number of data points is rather low. Molecular structures are complex graphs, which makes it hard for any model to derive a sufficient amount of information from this dataset alone. Second, the information available in the dataset is not complete: a substantial amount of toxicity information is missing in this dataset. There are 4,752 molecules for which at least one of the 12 kinds of toxicity is not known. In the prior literature, this issue has been approached in different ways. Some approaches perform data cleaning which limits the number of available data points even further, e.g. [136] excluded all data points that had any missing labels. We decided to keep these data points as part of our dataset but exclude the missing labels from the calculation of all loss functions and metrics. Any outputs that the model generates for these missing labels are considered correct and do not influence the training gradient. This approach allows the network to fill these gaps autonomously.

5.1.2 Methods

Figure 5.5 depicts the training setup that we used to evaluate the effects of ontology pre-training. This resulted in a three-step process. As the first step, we pre-trained an Electra model for 100 epochs on the *Mol-Pretrain* dataset described in 4.1.3. Afterward, we created two copies of this model, one of which was then trained further on the ontology expansion task using the $\text{ChEBI}^{854}_{v200}$ dataset and the same hyperparameters as reported in Section 4.4.2. Therefore, there are two models: One that received the ontology pre-training and one that did not. We then copy each of these models again and train them on either the Tox21 dataset used in the competition or the one published by MoleculeNet.

Preliminary results on the validation set showed that indicated that the models tend to overfit when trained based on the same hyperparameters as in Table 4.1 in Section 2.4.8. We, therefore, manually tuned the hyperparameters of the model and found that strong regularization impacted the training positively. The final experiments used the Adamax optimizer dropouts on hidden states of 0.4 and L2-regularisation of 0.0001. In order to address the relatively small amount of data and the thereby induced lack of variance, we also implemented an additional dropout directly on the input vectors, that hides random parts of the molecule from the model. Each input token had a probability of 20% to be hidden by this input dropout.

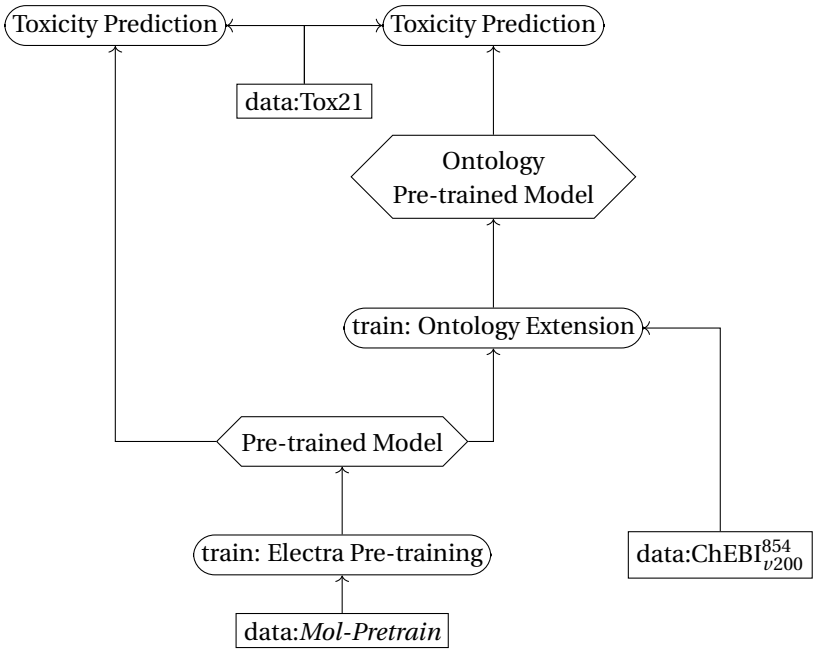


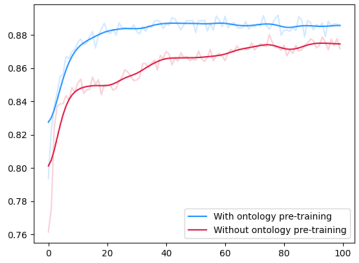
Figure 5.5: Training stack for standard training and ontology pre-training depicted using a simplified boxology notation [137] for one of the Tox21 datasets.

5.1.3 Results

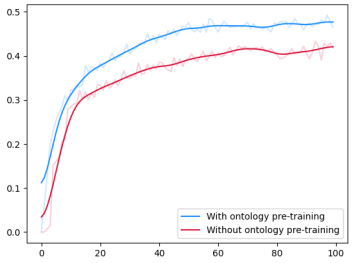
The final results of our training stack are four models: with or without ontology pre-training and fine-tuned on the original Tox21 competition dataset or on the smaller version of the Tox21 dataset published as part of MoleculeNet. The semantic pre-training already showed a clear impact during the training phase. Figures 5.6a-5.6d depict the curves for two metrics (F1 score and ROC-AUC) on our validation set as evaluated at the end of each epoch during training. We also included the performance reported for the recently published SSL-GCN model [138], which is trained by co-training two Graph Convolutional Networks using the SSL "mean teacher" method [139], which outperformed the successful DeepChem [76] models.

It can be seen that models with ontology pre-training start with better initial performance and also retain this margin throughout the training. This behavior is also reflected in the predictive performance on both test sets. Table 5.1 shows the predictive behavior for the dataset from MoleculeNet and the original challenge. The leading model (highlighted in bold) is predominantly the one that received additional ontology pre-training. This is particularly visible for the more noisy and sparse dataset used in the original Tox21 competition. The overall improved performance shows that pre-training with a more general ontology pre-training supports the network for the more specialized

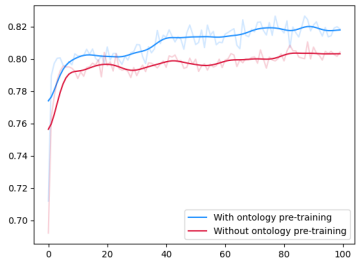
toxicity prediction. The drastic drop that can be seen around epoch 50 in Figure 5.6d but not for the pre-trained model in Figure 5.6b further indicates that ontology pre-training hedges the model against early overfitting. The reported results, and in particular the F1 scores, however, show that there is still a large margin of error for this task.



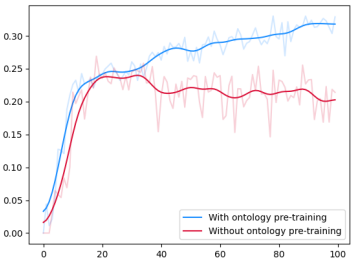
(a) ROC-AUC for the MoleculeNet Tox21 dataset



(b) F1 score (micro) for the MoleculeNet Tox21 dataset



(c) ROC-AUC for the original TOX21 dataset



(d) F1 score (micro) for the original TOX21 dataset

Figure 5.6: Development of ROC-AUC and F1 score (micro) during training on the validation sets of the Tox21 dataset available as part of MoleculeNet and the original TOX21 challenge.

In this section, we presented a novel approach for pre-training based on ontologies with structural annotations. Our analysis indicates that this pre-training can aid the fine-tuning on tasks that are related to the domain covered by the ontology even if the terminology for this task is not explicitly represented in the ontology. Usual pre-training approaches are based on masking tasks that hide parts of the input string. The model then has to predict the token that was masked. In the case of Electra, there is an additional layer that, given the input string in which the masked token has been replaced by the first models' prediction, has to predict which token of the input sequence has been masked. These approaches, however, can only cover syntactic representation. A model that has been pre-trained on a mask language task for natural

Dataset	Tox 21 (MoleculeNet)					Tox21 (Challenge)			
Metric	F1		ROC-AUC			F1		ROC-AUC	
Model	Our Model				SSL-GCN	Our Model			
Ontology Pre-training	yes	no	yes	no	-	yes	no	yes	no
NR-AR	0.41	**0.52**	**0.82**	0.76	0.80	0.1	**0.14**	**0.63**	0.62
NR-AR-LBD	**0.51**	0.5	**0.85**	0.77	0.76	0.05	**0.1**	**0.69**	0.67
NR-AhR	**0.53**	0.45	0.81	0.82	**0.83**	**0.23**	0.05	**0.8**	0.69
NR-Aromatase	**0.33**	0.15	**0.84**	0.8	0.73	**0.25**	0.04	**0.75**	0.69
NR-ER	**0.44**	0.4	**0.74**	0.71	0.72	**0.16**	0.09	**0.64**	0.62
NR-ER-LBD	**0.37**	0.3	**0.84**	0.76	0.69	**0.14**	0.12	**0.66**	0.63
NR-PPAR-gamma	**0.29**	-	**0.84**	0.83	0.76	**0.14**	-	**0.67**	0.66
SR-ARE	0.48	**0.53**	0.8	**0.84**	0.73	**0.37**	0.23	**0.71**	0.69
SR-ATAD5	0.14	**0.19**	**0.75**	0.74	0.72	**0.16**	-	**0.65**	0.65
SR-HSE	**0.24**	0.22	0.82	**0.82**	0.78	**0.13**	0.09	**0.76**	0.68
SR-MMP	**0.62**	0.53	**0.9**	0.88	0.81	**0.48**	0.21	**0.86**	0.82
SR-p53	**0.39**	0.35	**0.83**	0.8	0.75	**0.3**	-	**0.82**	0.78

Table 5.1: Class-wise scores on the test set on both Tox21 datasets. Bold values denote the best value for a particular combination of dataset and metric. NR - nuclear receptor; AR - androgen receptor; LBD - luciferase; AhR - aryl hydrocarbon receptor; ER - estrogen receptor; PPAR - peroxisome proliferator-activated receptor; SR - stress response; ARE - nuclear factor antioxidant response; ATAD5 - genotoxicity; HSE - heat shock factor response; MMP - mitochondrial response; p53 - DNA damage response.

language may be able to correctly predict in the string "Our [MASK] barks at the mailman", the masked token was probably "dog". But this prediction would solely be based on the fact that these words are likely to occur in this arrangement. It does not learn the semantics of these words or the target domain at large. Ontologies, on the other hand, are designed by domain experts to represent a concrete domain with fixed intended semantics. The ontology pre-training presented in this section therefore goes beyond the traditional syntactical pre-training in that it grants the model access to this source of knowledge.

The expert knowledge that is contained in structural annotations can then be used as a foundation to accelerate training tasks for a variety of domain-related tasks. Additionally, as results in Table 5.1 show, we were able to improve the performance of our model for toxicity prediction with the help of ontology pre-training. Further, as we will discuss in the next section, the performance of the ontology pre-trained model compares favorably with the state of the art.

The ontology pre-training approach presented in this section can be applied to any problem in which there are structural annotations from the ontology that have the same structure as the input features of the problem. But this is not a necessary requirement. It is plausible that this same procedure could hypothetically be used for any arbitrary dataset and ontologies, given a method that links its data points to sets of classes within the domain ontology in the

same way that ChEBI's annotations link SMILES strings to ontology classes. This mapping could then be used to turn these data points into artificial annotations that can be used for the ontology extension task. The feasibility of this approach, however, depends on the quality of this mapping and the quality of the ontology in general.

5.1.4 Discussion

In this section, we presented an approach that allows the inclusion of an ontology into the learning process. By introducing an additional ontology-based pre-training step, we enabled the system to improve the predictive performance when fine-tuned on the task of toxicity prediction. This implied that, even though the concept of toxicity was not explicitly represented in the fragment of the ontology used in this analysis, the model was still able to extract relevant domain knowledge that allowed it to solve this task. ChEBI contains chemical, biological or pharmacological roles. We are currently investigating the effect of including these parts of the ontology on the task of toxicity prediction. Another open question is the exact impact of the ontological hierarchy and the quality of the ontology. It seems reasonable to assume that an ontology of higher quality has a higher impact on the final prediction task than others. A low-quality ontology might even have a detrimental effect as it causes the model to focus on structures of artifacts that are not relevant to the domain. While we limited this analysis to only this particular use case, the technique can be applied to arbitrary classification tasks that are based on structural information of chemical compounds. The field of application can be extended even further by performing the pre-training on different ontologies.

5.2 Semantic Loss

My contributions

The following section is currently unpublished work. The conceptualization, methods and analyses in this section have been developed and analyzed by myself with input regarding the formulation of the different measures of inconsistency by Fabian Neuhaus. I also authored a related paper [140] that extends the results presented here and is currently under review.

The labeling of things, e.g. "this is a bird", is a fundamental technique used by parents to teach general concepts to children [141]. This method is akin to the

way we trained machine learning models in Chapter 4. We used the ontology and its subsumption axioms in particular to derive the appropriate labels for molecules and used those to teach the system. This method, however, only gives the system indirect access to these axioms. A direct integration of these axioms, e.g. "penguins are birds" is not available during training. Additionally, this utilizes only a single kind of axiom, while most ontologies cover a wider variety, such as disjointness axioms. These allow the expression of "is-not-a" relations, e.g. "A bird is not a mammal." In summary, two things are desirable: The usage of more ontological axioms and a more explicit involvement of logical axioms during training and evaluation. In this section, we would like to present a method with which the axioms of an ontology can be directly integrated into the learning process.

5.2.1 Methods

We exploit the fact that from $A \sqsubseteq B$ follows that $\forall x : A(x) \rightarrow B(x)$. In boolean logic, an implication $a \rightarrow c$ is false of the antecedent a is true and the consequence c is false. Yet, we consider the subsumption relationships in the given ontology to be correct. Predictions, for which the respective implication would be false, are considered violations. The membership predictions of our models are values in $[0,1]$. In our evaluations in Chapter 4, we interpreted a predicted membership of more than 0.5 as a positive prediction for that class. Using this threshold, we can count the boolean implication violations:

Definition 4 (Boolean Violation) *Let C be a set of classes and $R \subseteq C \times C$ a set of subsumption relations over this set. Let further be $h_c : c \rightarrow [0,1]$ be a fuzzy membership function for a given class $c \in C$. We define the degree of violation in boolean logic as ϕ^R_{binary} for subsumptions over R as*

$$\phi^{\sqsubseteq}_{binary}(x) := |\{(l,r) \in R : h_l(x) > 0.5 \wedge h_r(x) \leq 0.5\}| \tag{5.1}$$

Note that the boolean violation metric is equivalent to the FNR metric used in [140]. The most successful approaches from Chapter 4 are all based on gradient descent. For a given prediction, the gradient of the objective function is calculated to determine the direction in which the hyperparameters need to be adjusted. In order to define a corresponding penalty term for logical inconsistency, we therefore need a function that is at least partially differentiable. For this we use the implication of Łukasiewicz fuzzy logic $\hat{h}(a \rightarrow b) = \min(1, 1 - \hat{h}(a) + \hat{h}(b))$, or more precisely its negation $\hat{h}(\neg(a \rightarrow b)) = \max(0, \hat{h}(a) - \hat{h}(b))$:

Definition 5 (Łukasiewicz Violation) *Let C be a set of classes and $R \subseteq C \times C$ a set of subsumption relations over this set. Let further be $h_c : c \rightarrow [0,1]$ be a fuzzy*

membership function for a given class $c \in C$. We define the degree of violation ϕ^R_{luka} for subsumptions over R as

$$\phi^{\sqsubseteq}_{luka}(x) := \sum_{(l,r)\in R} \max(h_l(x) - h_r(x), 0) \tag{5.2}$$

The Łukasiewicz violation metric aggregates violations based on their degree. In order to derive the absolute value of violations, we also count the occurrence of violations regardless of degree, i.e. all instances in which the predicted membership value for the class on the left-hand side of a subsumption relation is larger than the predicted membership for that on the right-hand side. This heavily penalizes even small violations. Machine learning systems are prone to produce noisy predictions. This noise is, however, an inherent property of the gradient descent used during training and not a sign that the system does not correctly represent the intended subsumption relation. Therefore, we also introduce a weakened version of this metric that allows for violations of at most 0.01.

Definition 6 (Strong and Weak Gaines-Rescher Violation) *Let C be a set of classes and $R \subseteq C \times C$ a set of subsumption relations over this set. Let D be a domain and $h_c : D \to [0,1]$ a fuzzy membership function for a given class $c \in C$. We define the degree of violation ϕ^R_{luka} for subsumptions over R as*

$$\phi^{\sqsubseteq}_{strict}(x) := |\{(l,r) \in R : h_l(x) - h_r(x) > 0\}| \tag{5.3}$$

$$\phi^{\sqsubseteq}_{weak}(x) := |\{(l,r) \in r : h_l(x) - h_r(x) > 0.01\}| \tag{5.4}$$

Two classes A, B are disjoint iff $A \sqsubseteq \neg B$. Therefore, we can define analogous functions $\phi^{\not\sqsubseteq}_{luka}$, $\phi^{\not\sqsubseteq}_{binary}$, $\phi^{\not\sqsubseteq}_{strict}$, $\phi^{\not\sqsubseteq}_{weak}$ with $h(\neg B) = 1 - h(B)$. Both $\phi^{\sqsubseteq}_{luka}$ and $\phi^{\not\sqsubseteq}_{luka}$ are used as penalties for the loss during training.

With this setup, we repeat our task for ontology extension from Chapter 4 on the $\text{ChEBI}^{854}_{v200}$ dataset. We add the terms for implication $\text{softmax}(\phi^{\sqsubseteq}_{luka}) \cdot \phi^{\sqsubseteq}_{luka}$ and disjointness $\text{softmax}(\phi^{\not\sqsubseteq}_{luka}) \cdot \phi^{\not\sqsubseteq}_{luka}$ as an additional penalty term to the binary cross-entropy loss. Preliminary tests have shown that this scaling prevents an exploding gradient that led to a drop in performance for models using the Łukasiewicz violation [140]. There are some methodological differences between the results presented here and in [140]. First, as for the rest of this thesis, we use ChEBI version 200 and models have been trained for 100 epochs and with softmax scaling, while [140] uses the more recent version 231 and models that have been trained for 200 epochs without softmax scaling.

Xu et al. [142] proposed a more general definition of a semantic loss for arbitrary logical sentences. The semantic loss presented in this section can be

seen as a special case of this formalization. An implication loss would be defined in their formulation for a specific axiom $A \sqsubseteq B$ and a prediction vector p over all satisfying models x:

$$\begin{aligned} L^s(A \sqsubseteq B, p) &\propto -\log \sum_{x \models A \sqsubseteq B} \prod_{x \models X_i} p_i \prod_{i : x \models \neg X_i} (1 - p_i) \\ &= -\log(p_a \cdot p_b - p_a + 1) \\ &= -\log(1 - p_a \cdot (1 - p_b)) \end{aligned}$$

This loss definition is therefore related to the one used in this work, but based on the product t-norm $T(a,b) = a \cdot b$ instead of the Łukasiewicz t-norm used in this work. A further investigation of this metric in [140] shows that this metric performs similarly to other definitions of fuzzy conjunction.

Training with unlabeled data

Humans are able to verify statements like "I saw an animal that was a cat and a dog" without actually knowing the entity that is referred to. This is possible due to the background knowledge that the concept of "cat" is disjoint from the concept of "dog". A common instance can therefore not exist. Similarly, a statement like "I saw a dog that was not a mammal" is false, regardless of the entity in question. Likewise, the semantic loss allows the system to evaluate the logical consistency of a prediction without knowing the actual ground truth.

We also conducted the fine-tuning of our system on a dataset that did not only include the $\text{ChEBI}^{854}_{v200}$ dataset but also a mix-in of the same amount of unlabeled data randomly sampled from Pubchem. The model is trained on the same ontology extension task that we discussed in Chapter 4. The loss function was calculated as before for those instances from the $\text{ChEBI}^{854}_{v200}$. For the new, unlabeled fragment, we did not calculate the binary cross-entropy loss and instead based the loss solely on the violations for implication $\phi^{\sqsubseteq}_{luka}$ and disjointness $\phi^{\not\sqcap}_{luka}$. We ensured for this extension that the training, validation and test splits were extensions of their counterparts from the $\text{ChEBI}^{854}_{v200}$ dataset in order to ensure comparability.

5.2.2 Results

We evaluated the model that was trained without semantic loss with the two models that used this additional penalty on the two data sets that were also used in the training: The $\text{ChEBI}^{854}_{v200}$ dataset and the extended variant.

Table 5.2 shows the results of the different implication violation metrics for labeled and PubChem data. Both models that were trained with semantic loss outperformed the standard model. The model that has been trained using

	$\phi^{\sqsubseteq}_{binary}$	$\phi^{\sqsubseteq}_{luka}$	$\phi^{\sqsubseteq}_{strict}$	$\phi^{\sqsubseteq}_{weak}$
baseline	3.3253e-5	3.8132e-5	*0.1303*	0.00046
semantic loss	**2.1481e-5**	**2.6676e-5**	**0.1071**	**0.00040**
semantic loss (mixed)	*2.9445e-5*	*3.3529e-5*	0.1465	*0.00044*

Table 5.2: Implication violations evaluated on labeled data only, averaged over all instances.

	$\phi^{\sqsubseteq}_{binary}$	$\phi^{\sqsubseteq}_{luka}$	$\phi^{\sqsubseteq}_{strict}$	$\phi^{\sqsubseteq}_{weak}$
baseline	1.2849e-05	2.1121e-05	**0.0502**	0.0003
semantic loss	*8.8642e-06*	*1.1136e-05*	0.1764	*0.00016*
semantic loss (mixed)	**8.0575e-06**	**1.0411e-05**	*0.1422*	**0.00018**

Table 5.3: Implication violations evaluated on unlabeled PubChem data averaged over all instances.

semantic loss with labeled data showed better performance for all metrics when evaluated on the labeled test dataset. The model that was trained on mixed data performed better in the evaluation of the unlabeled PubChem set. This indicates that the additional training is hedging the model against out-of-distribution data, but at the same time, introduces noise into the training, which worsens its performance on labeled data.

	$\phi^{\not\sqsubseteq}_{binary}$	$\phi^{\not\sqsubseteq}_{luka}$	$\phi^{\not\sqsubseteq}_{strict}$	$\phi^{\not\sqsubseteq}_{weak}$
baseline	5.9851e-7	6.3729e-7	0.0010	5.1692e-6
semantic loss	**4.8858e-9**	**1.3880e-8**	*0.0005*	**2.2475e-7**
semantic loss (mixed)	*9.7716e-9*	*2.1747e-8*	**0.0002**	*2.5162e-07*

Table 5.4: Disjointness violations evaluated on labeled data only, averaged over all instances.

A similar pattern emerges for the analysis of the disjointness loss on labeled data in Table 5.4 and unlabelled data in Table 5.5. The overall values are much smaller, which indicates that the models can successfully learn to separate classes that do not have any mutual members. Remarkably, the mixed model caused no violation on the unlabelled dataset, but all models caused very few violations of disjointness axioms. In particular, no model that was trained with a semantic loss caused violations on the $\phi^{\not\sqsubseteq}_{binary}$ metric, i.e. none of them made predictions that, after applying the threshold, violate a disjointness axiom. It can be seen in all these results, that the strict metric is usually an outlier when it comes to performance quality. This indicates that a vast majority of violations are on a very small margin. This means that the membership values for the left and the right side of the implications are close to each other but violate the constraints due to instabilities. Such violations contribute the all

	$\phi^{\not\sqsubseteq}_{binary}$	$\phi^{\not\sqsubseteq}_{luka}$	$\phi^{\not\sqsubseteq}_{strict}$	$\phi^{\not\sqsubseteq}_{weak}$
baseline	9.7716e-09	2.174e-08	0.0003	2.5162e-07
semantic loss	**0.0**	*1.3478e-10*	*1.4544e-07*	*9.4854e-09*
semantic loss (mixed)	**0.0**	**0.0**	**0.0**	**0.0**

Table 5.5: Disjointness violations evaluated on unlabeled PubChem data averaged over all instances.

other loss definitions only marginally but are counted as full violations under the strict metric.

	Micro-F1	Macro-F1
baseline	**0.9032**	**0.6082**
semantic loss	0.8939	0.5934
semantic loss (mixed)	0.8843	0.5200

Table 5.6: F1-scores calculated on labeled data.

Lastly, we also evaluated the predictive performance of all three models. As can be seen in Table 5.6, the macro F1-scores for the models that were trained with semantic loss, are generally worse than for the one that did not get this additional penalty. This is particularly true for the F1-Score with macro aggregation for the model that was trained on mixed data.

Our results indicate that the introduction of a semantic loss during training increases the overall logical consistency of predictions. These results are consistent with results from [140]. This increase in consistency, however, comes at the detriment of the actual predictive quality. This result seems contradictory at first, as one would assume more consistent results to be better overall. This behavior may be attributed to the data imbalance that is inherently introduced by the hierarchical structure of the ontology. Classes that are further down in the hierarchy have fewer members and are therefore underrepresented in comparison to those higher up in the hierarchy. The gradient of the implication loss is

$$\frac{\partial \phi^{\sqsubseteq}_{luka}}{\partial \theta}(x) = \sum_{(l,r)\in R} \begin{cases} \frac{\partial h_l}{\partial \theta}(x) - \frac{\partial h_r}{\partial \theta}(x) & \text{if } h_l(x) > h_r(x) \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

For a violation of an axiom $A \sqsubseteq B$, this term can be minimized by increasing the predictions of B or decreasing the one of A. This puts an additional incentive on the system to not predict smaller classes, which are already under-represented in the dataset. The system therefore tends to favour larger classes over smaller ones, which is also supported by the drop in macro F1 for the models that use semantic loss. Further analysis of this behavior in [140]

showed that an asymmetric loss function that skews the gradient towards the subclass can mediate this tendency.

The results also indicate that the inclusion of unlabelled data into the training process hedges the system against inconsistencies when confronted with unseen data. This result can be particularly useful in scenarios in which the distribution of features in the dataset is limited. Deep learning systems are prone to suffer from out-of-distribution errors, e.g. unpredictable behavior on data that has not been sampled from the same distribution as the training data. Lifting this limitation is often not easy because additional labeling is required. The semi-supervised training method presented here can help to alleviate this problem.

5.2.3 Discussion

In this section, we have presented an approach to directly integrate an ontology into the training process. By evaluating the prediction of a model for logical consistency with respect to the ontology, we were able to define an additional loss penalty for two different kinds of axioms. We defined four different notions of violations for subsumption and disjointness axioms. Our analysis indicates that models, that were trained using this additional penalty, produce more logically consistent predictions and are therefore more reliable on unseen and out-of-distribution data. The analysis has, however, also shown that the introduction of this additional penalty has a negative impact on the overall model performance for smaller ontology classes. While this difference is marginal when training on labeled data only, the impact of mixed training is particularly severe but also improves the logical consistency for out-of-distribution predictions. This method can be extended to arbitrary ontology axioms, such as more complex class expressions such as $A \sqcap B \sqsubseteq C$. Therefore, this method allows for a flexible inclusion of ontology axioms into the training process.

A comparison with [140] shows that the results presented here can be further improved with a more recent version of ChEBI and longer training. The improved performance on a newer version of ChEBI is particularly notable because this allows for a frictionless upgrade of systems such as Chebifier to newer versions and therefore a continuous evolution of the system alongside ChEBI.

6

Conclusions and Future Work

Ontologies and other semantic technologies provide an important basis for modern knowledge representation. However, the development of these usually requires extensive expert knowledge, a high degree of domain-specific expertise and, above all, time. At the same time, once they have been created, they offer a rich pool of expert knowledge that is expressed not only in the explicitly represented facts and axioms but also in the way they are structured.

Machine learning methods, on the other hand, are often very flexible and can be applied to a variety of problems in a wide range of domains. However, they often suffer from a lack of interpretability and the logical consistency of results - especially on unseen data - cannot be guaranteed.

In this work, we have presented methods with which these two worlds can be brought closer together. We have seen how machine learning can be used to support ontology development. We have used special structural annotations to train a system on a given domain ontology. We evaluated the functionality of this model using the ChEBI ontology. The system we trained not only showed qualitatively better classifications than the ClassyFire tool currently used in ChEBI development but it is also flexibly adaptable to new versions of ChEBI. This allows an iterative process in which the ChEBI ontology is extended with predictions from our system, this extension is then corrected by experts and used for a new training of our system. With Chebifier, we have also developed an interface that allows users to utilize our prediction model smoothly. This removes the barrier to using our tool and also allows it to be used beyond ChEBI. We, therefore, hope that Chebifier will be integrated into ChEBI's development process. The ChEBI ontology itself is also used in numerous practical areas. These can also benefit from this new classification tool.

We have also shown that the semantic structure of ontologies can be used to support a machine-learning system. Here, an additional ontology-based training step could be used to predict the properties of molecules that were

not present in the training fragment. The expert knowledge that went into structuring the ontology was able to provide crucial information that allow the model to learn in a more focused way. Furthermore, we were able to use the logical axioms of ontology to train a machine learning system so that the predictions are logically more consistent.

6.1 Future Work

We have demonstrated different types of synergy between ontologies and machine learning. However, in our opinion, the possibilities of these methods are far from exhausted. Finally, we would like to point out which open questions and possible directions for further development we currently see.

In our ontology expansion system, we basically used a two-layer system in which data is first encoded and then passed to a prediction model. Each of these components allows for a variety of possible improvements. The database we used was extracted from a fixed version of ChEBI. Since then, new enhancements have been made to ChEBI that are not yet represented in the current model. On the one hand, this would further increase the number of available data points. However, due to our selection method, this would also mean that more classes would pass the thresholds we have set. As a result, the prediction model would also have a larger number of target classes. However, our analyses have also shown that although ChEBI grows comparatively quickly, this growth mainly takes place in the area of not fully annotated 2-star molecules. The 3-star fragment grows comparatively slowly. It would therefore also be conceivable to carry out the training in the future only on the 3-star fragment of ChEBI. Although this contains less data, it is of high quality and has been annotated by experts. But even with the data that we currently use, there may still be ways to improve the performance. For one, we do not yet incorporate any stratification mechanisms to ensure an even distribution of classes in training, test and validation set. Furthermore, we are mostly using SMILES strings as they are used in ChEBI. To our knowledge, these SMILES strings are generated using RDKit, which can produce canonized SMILES strings. There are many ways to represent the same molecule in SMILES. A more varied representation of molecules may allow for the generation of more training data and a less biased model.

In the encoding layer of our system, we currently primarily utilize SMILES strings with the SMILES-based tokenizer. Additional encoding methods could be employed here. With DeepSmiles and SELFIES, there are already two encodings specifically designed for certain applications in deep learning. Furthermore, our Chem Tokenizer can also be combined with the BPE (Byte Pair

Encoding) method to generate chemically meaningful token groups. This can be particularly helpful in analyzing large molecules or searching for functional groups. Additionally, we are currently not incorporating any supplementary information. Considering that ontological distinctions in ChEBI are made based not only on structural but also functional properties or chemical origin, the enrichment of input data could lead to a significant improvement.

The fragment of ChEBI that we used was also limited in certain respects. As we argue in Chapter 4, a further extension of the covered classes by way of lowering the threshold for class selection is not promising. There are, however, further valuable sources of domain knowledge that may be integrated in different parts of our system. One example here is the role hierarchy in ChEBI. These roles include different information about compounds, e.g. that they can function as a nutrient or toxin. This is of particular interest for the ontology pre-training, as it may convey more information to the system and, thereby, allow for the extraction of more domain knowledge.

Most of our approaches were based on a transformer-based model. While these models are quite potent, they still come with unique downsides. We have demonstrated that an analysis of attention weights can give some insights into the model's inner workings, but this is still far from actual explainability. This is a particularly important issue in an area of application in which ontology developers might want to know why the model reached a certain (mis-)classification for a given compound. Therefore, we aim to integrate architectures that allow for more transparency. Graph neural networks (GNN) in particular, are promising in two regards. First, they allow for the direct representation of a molecular graph without the transformation into a sequence. This way, cycles and branches can be presented without the need to break cycles and branches. At the same time, there is a close relationship to formal methods. Grohe showed that the set of queries that can be answered by bounded graph neural networks is equivalent to those definable a particular fragment of first-order logic [143], a further investigation of this connection may lead to more explainable methods. Another promising approach are Logical Neural Networks (LNN), which we already discussed briefly in Section 3.4, which allow for a direct representation of a logical theory such as an ontology. But they can also be applied directly to molecular structures. We already successfully used first-order reasoning for the classification of chemicals [66]. While this logical framework may not be sufficient to respresent all chemical structures [26], it may still cover a significant fraction of ChEBI.

The investigation of the specific properties of the ontology-based semantic loss is still ongoing research. We aim to investigate the influence of different T-norms and the ways semantic loss formulations can be used in combination with different machine-learning architectures.

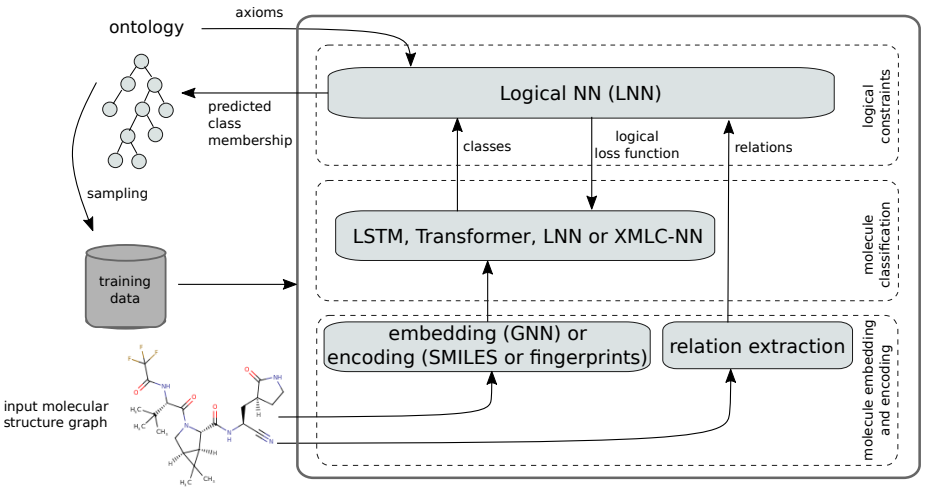


Figure 6.1: Ontology extension system as envisioned in StrOntEx

Parts of these points are currently being investigated in our new research project *Structured-entities Ontology Extension* (StrOntEx, funded by German Research Foundation - DFG - project number 522907718) that has been proposed based on the work presented in this thesis. Within this project, the presented approach for ontology is going to be part of a larger system that covers a larger number of approaches that interact in various ways as depicted in Figure 6.1. The basic working principle for the first two layers remains as described in Chapter 4. However, the transformer model will be part of a larger ensemble of different approaches that cover different parts of the domain. In addition to the subsumption hierarchy of the ontology, we will also include relations and chemical properties in the dataset but the overall system presented in this work will be extended by an additional layer that directly represents the logical structure of the ontology. Herein it is important that the chosen structure can adequately represent the rich semantics of OWL or even stronger logics. Here we may choose from a number of different neuro-symbolic architectures, such as Logical Neural Networks [97], ELEmbeddings [102] or Logic Tensor Networks [94]. The last two layers of our architecture are not just a sequential bottom-up model. Instead, we will also implement a semantic downwards path from the logic layer back to the prediction layer - not unlike the semantic loss presented in Section 5.2. With this architecture, we are positive that we can build on the successful work presented in this thesis, support ontology developers in their important work and further the development of systems that integrate machine learning and symbolic methods.

A

Bibliography

[1] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

[2] Meisam Booshehri, Lukas Emele, Simon Flügel, Hannah Förster, Johannes Frey, Ulrich Frey, Martin Glauer, Janna Hastings, Christian Hofmann, Carsten Hoyer-Klick, et al. Introducing the open energy ontology: Enhancing data interpretation and interfacing in energy systems analysis. *Energy and AI*, 5:100074, 2021.

[3] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[4] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[5] Gene Ontology Consortium. The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, 47(D1):D330–D338, 2019.

[6] Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36 (1):D344–D350, 2007.

[7] Janna Hastings, Paula De Matos, Adriano Dekker, Marcus Ennis, Bhavana Harsha, Namrata Kale, Venkatesh Muthukrishnan, Gareth Owen, Steve Turner, Mark Williams, et al. The chebi reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic acids research*, 41(D1):D456–D463, 2012.

[8] Janna Hastings, Gareth Owen, Adriano Dekker, Marcus Ennis, Namrata Kale, Venkatesh Muthukrishnan, Steve Turner, Neil Swainston, Pedro Mendes, and Christoph Steinbeck. Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic acids research*, 44 (D1):D1214–D1219, January 2016. ISSN 1362-4962. doi: 10.1093/nar/gkv1031.

[9] ChEBI Statistics. `https://www.ebi.ac.uk/chebi/statisticsForward.do`, 2024. Accessed: 2024-01-25.

[10] National Library of Medicine – National Center for Biotechnology Information. `https://www.ncbi.nlm.nih.gov/pccompound/?term=(%222023%2F1%2F01%22%5BCreateDate%5D+%3A+%222023%2F1%2F31%22%5BCreateDate%5D)`, 2024. Accessed: 2024-01-25.

[11] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2023 update. *Nucleic Acids Res.*, 51(D1):D1373–D1380, January 2023. ISSN 0305-1048. doi: 10.1093/nar/gkac956. URL `https://doi.org/10.1093/nar/gkac956`.

[12] Donna Medeiros, Donna White, and Kevin Gabbard. Hype Cycle for Data and Analytics Programs and Practices, 2023. `https://www.gartner.com/document/4597499`, 08 2023.

[13] Tom Gruber. What is an ontology, 1993.

[14] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25 (1-2):161–197, 1998.

[15] Fabian Neuhaus. On the definition of'ontology'. In *JOWO*, 2017.

[16] Fabian Neuhaus. What is an ontology? *arXiv preprint arXiv:1810.09171*, 2018.

[17] Darren A Natale, Cecilia N Arighi, Judith A Blake, Jonathan Bona, Chuming Chen, Sheng-Chih Chen, Karen R Christie, Julie Cowart, Peter

D'Eustachio, Alexander D Diehl, et al. Protein ontology (pro): enhancing and scaling up the representation of protein entities. *Nucleic acids research*, 45(D1):D339–D346, 2017.

[18] Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, et al. The gene ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, 2023.

[19] Rebecca Jackson, Nicolas Matentzoglu, James A Overton, Randi Vita, James P Balhoff, Pier Luigi Buttigieg, Seth Carbon, Melanie Courtot, Alexander D Diehl, Damion M Dooley, William D Duncan, Nomi L Harris, Melissa A Haendel, Suzanna E Lewis, Darren A Natale, David Osumi-Sutherland, Alan Ruttenberg, Lynn M Schriml, Barry Smith, Christian J Stoeckert Jr., Nicole A Vasilevsky, Ramona L Walls, Jie Zheng, Christopher J Mungall, and Bjoern Peters. OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. *Database*, 2021: baab069, 10 2021. ISSN 1758-0463. doi: 10.1093/database/baab069. URL https://doi.org/10.1093/database/baab069.

[20] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[21] OpenSMILES specification — opensmiles.org. http://opensmiles.org/opensmiles.html, 2024. [Accessed 31-01-2024].

[22] Harry L Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113, 1965.

[23] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[24] Greg Landrum. RDKit: Open-source cheminformatics, 2020. URL https://www.rdkit.org/.

[25] Yannick Djoumbou Feunang, Roman Eisner, Craig Knox, Leonid Chepelev, Janna Hastings, Gareth Owen, Eoin Fahy, Christoph Steinbeck, Shankar Subramanian, Evan Bolton, Russell Greiner, and David S. Wishart. ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. *Journal of Cheminformatics*, 8(1): 61, December 2016. ISSN 1758-2946. doi: 10.1186/s13321-016-0174-y. URL https://jcheminf.biomedcentral.com/articles/10.1186/s13321-016-0174-y.

[26] Oliver Kutz, Janna Hastings, and Till Mossakowski. Modelling Highly Symmetrical Molecules: Linking Ontologies and Graphs Artificial Intelligence: Methodology, Systems, and Applications. In Allan Ramsay and Gennady Agre, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, volume 7557 of *Lecture Notes in Computer Science*, pages 103–111. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33184-8. doi: 10.1007/978-3-642-33185-5_11. URL `http://dx.doi.org/10.1007/978-3-642-33185-5_11`. Section: 11.

[27] David S Wishart, Sagan Girod, Harrison Peters, Eponine Oler, Juan Jovel, Zachary Budinski, Ralph Milford, Vicki W Lui, Zinat Sayeeda, Robert Mah, et al. Chemfont: the chemical functional ontology resource. *Nucleic Acids Research*, 51(D1):D1220–D1229, 2023.

[28] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[29] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[30] J Mercer. Functions of positive and negativetypeand their connection with theory ofintegral equations. *Philosophical Trinsoctions of Royal Society*, pages 4–415, 1909.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 30, 2017.

[33] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[34] Mohamad Ballout, Ulf Krumnack, Gunther Heidemann, and Kai-Uwe Kühnberger. Opening the black box: analyzing attention weights and hidden states in pre-trained language models for non-language tasks. In *World Conference on Explainable Artificial Intelligence*, pages 3–25. Springer, 2023.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[36] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[37] Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. Accurate computation of the log-sum-exp and softmax functions. *arXiv preprint arXiv:1909.03469*, 2019.

[38] Houssem Assadi. Construction of a Regional Ontology from Text and its Use within a Documentary System. In *FOIS'98 - 1st International conference on Formal Ontology in Information Systems*, volume 46 of *Frontiers in Artificial Intelligence and Applications*, pages 236–252, Trento, Italy, June 1998. IOS Press. URL `https://hal.archives-ouvertes.fr/hal-01617868`.

[39] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79, 2001.

[40] Chris Biemann. Ontology learning from text: A survey of methods. In *LDV forum*, volume 20, pages 75–93, 2005.

[41] Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. A survey of ontology learning techniques and applications. *Database*, 2018: bay101, 10 2018. ISSN 1758-0463. doi: 10.1093/database/bay101. URL `https://doi.org/10.1093/database/bay101`. bay101.

[42] Ana Ozaki. Learning description logic ontologies: Five approaches. where do they stand? *KI-Künstliche Intelligenz*, 34(3):317–327, 2020.

[43] Efthymios G Drymonas. Ontology learning from text based on multiword term concepts: the ontogain method. *Master of Science thesis, Technical University of Crete, Greece*, 2009.

[44] Euthymios Drymonas, Kalliopi Zervanou, and Euripides GM Petrakis. Unsupervised ontology acquisition from plain texts: the ontogain system. In *Natural Language Processing and Information Systems: 15th International Conference on Applications of Natural Language to Information Systems, NLDB 2010, Cardiff, UK, June 23-25, 2010. Proceedings 15*, pages 277–287. Springer, 2010.

[45] Roberto Navigli, Paola Velardi, and Aldo Gangemi. Ontology learning and its application to automated terminology translation. *IEEE Intelligent systems*, 18(1):22–31, 2003.

[46] Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213, 1999.

[47] Hermine Njike Fotzo and Patrick Gallinari. Learning" generalization/specialization" relations between concepts-application for automatically building thematic document hierarchies. In *RIAO*, volume 4, pages 143–155. Citeseer, 2004.

[48] Sara Althubaiti, Şenay Kafkas, Marwa Abdelhakim, and Robert Hoehndorf. Combining lexical and context features for automatic ontology extension. *Journal of biomedical semantics*, 11(1):1–13, 2020.

[49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[50] Licong Cui, Wei Zhu, Shiqiang Tao, James T Case, Olivier Bodenreider, and Guo-Qiang Zhang. Mining non-lattice subgraphs for detecting missing hierarchical relations and concepts in SNOMED CT. *Journal of the American Medical Informatics Association*, 24(4):788–798, 02 2017. ISSN 1067-5027. doi: 10.1093/jamia/ocw175. URL `https://doi.org/10.1093/jamia/ocw175`.

[51] Alina Petrova, Yue Ma, George Tsatsaronis, Maria Kissa, Felix Distel, Franz Baader, and Michael Schroeder. Formalizing biomedical concepts from textual definitions. *Journal of biomedical semantics*, 6(1): 1–17, 2015.

[52] Robert Arp, Barry Smith, and Andrew D Spear. *Building ontologies with basic formal ontology*. Mit Press, 2015.

[53] Stefano Borgo, Roberta Ferrario, Aldo Gangemi, Nicola Guarino, Claudio Masolo, Daniele Porello, Emilio M Sanfilippo, and Laure Vieu. Dolce: A descriptive ontology for linguistic and cognitive engineering. *Applied ontology*, 17(1):45–69, 2022.

[54] Alcides Lopes, Joel Carbonera, Daniela Schmidt, Luan Garcia, Fabricio Rodrigues, and Mara Abel. Using terms and informal definitions to classify domain entities into top-level ontology concepts: An approach based on language models. *Knowledge-Based Systems*, 265: 110385, 2023.

[55] Fabian Neuhaus and Janna Hastings. Ontology development is consensus creation, not (merely) representation. *Appl. Ontology*, 17(4):495–513, January 2022. ISSN 1570-5838. doi: 10.3233/AO-220273. URL `https://content.iospress.com/articles/applied-ontology/ao220273`. Publisher: IOS Press.

[56] Nele Köhler and Fabian Neuhaus. The mercurial top-level ontology of large language models, 2024.

[57] Pablo Moreno, Stephan Beisken, Bhavana Harsha, Venkatesh Muthukrishnan, Ilinca Tudose, Adriano Dekker, Stefanie Dornfeldt, Franziska Taruttis, Ivo Grosse, Janna Hastings, Steffen Neumann, and Christoph Steinbeck. BiNChE: a web tool and library for chemical enrichment analysis based on the ChEBI ontology. *BMC Bioinformatics*, 16:56, 2015. doi: 10.1186/s12859-015-0486-3.

[58] David P. Hill, Nico Adams, Mike Bada, Colin Batchelor, Tanya Z. Berardini, Heiko Dietze, Harold J. Drabkin, Marcus Ennis, Rebecca E. Foulger, Midori A. Harris, Janna Hastings, Namrata S. Kale, Paula de Matos, Christopher J. Mungall, Gareth Owen, Paola Roncaglia, Christoph Steinbeck, Steve Turner, and Jane Lomax. Dovetailing biology and chemistry: integrating the gene ontology with the chebi chemical ontology. *BMC genomics*, 14:513, July 2013. ISSN 1471-2164. doi: 10.1186/1471-2164-14-513.

[59] Gang Fu, Colin Batchelor, Michel Dumontier, Janna Hastings, Egon Willighagen, and Evan Bolton. PubChemRDF: towards the semantic annotation of PubChem compound and substance databases. *Journal of Cheminformatics*, 7:34, 2015. URL `https://doi.org/10.1186/s13321-015-0084-4`.

[60] María Herrero-Zazo, Isabel Segura-Bedmar, Janna Hastings, and Paloma Martínez. DINTO: Using OWL Ontologies and SWRL Rules to Infer Drug-Drug Interactions and Their Mechanisms. *Journal of chemical information and modeling*, 55(8):1698–1707, August 2015. ISSN 1549-960X. doi: 10.1021/acs.jcim.5b00119. URL `https://doi.org/10.1021/acs.jcim.5b00119`.

[61] Janna Hastings, Despoina Magka, Colin Batchelor, Lian Duan, Robert Stevens, Marcus Ennis, and Christoph Steinbeck. Structure-based classification and ontology in chemistry. *Journal of cheminformatics*, 4:8, April 2012. ISSN 1758-2946. doi: 10.1186/1758-2946-4-8.

[62] Despoina Magka, Markus Krötzsch, and Ian Horrocks. A rule-based ontological framework for the classification of molecules. *Journal of Biomedical Semantics*, 5(1):17, April 2014. ISSN 2041-1480. doi: 10.1186/2041-1480-5-17.

[63] Janna Hastings, Michel Dumontier, Duncan Hull, Matthew Horridge, Christoph Steinbeck, Ulrike Sattler, Robert Stevens, Tertia Hörne, and Katarina Britz. Representing chemicals using OWL, description graphs and rules. In *Proc. of OWL: Experiences and Directions (OWLED 2010)*, 2010.

[64] Despoina Magka, Boris Motik, and Ian Horrocks. Modelling Structured Domains Using Description Graphs and Logic Programming. In David Hutchison et al., editors, *The Semantic Web: Research and Applications*, volume 7295 of *Lecture Notes in Computer Science*, pages 330–344. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-30283-1. doi: 10.1007/978-3-642-30284-8_29.

[65] Claudia Bobach, Timo Böhme, Ulf Laube, Anett Püschel, and Lutz Weber. Automated compound classification using a chemical ontology. *Journal of Cheminformatics*, 4(1):1–12, 2012.

[66] Simon Flügel, Martin Glauer, Fabian Neuhaus, and Janna Hastings. When one logic is not enough: Integrating first-order annotations in OWL ontologies. *Semantic web journal*, 2023.

[67] Heinz-Dieter Ebbinghaus and Jörg Flum. Finite model theory, enlarged edn. *Springer Monographs in Mathematics. Springer-Verlag, Berlin*, 2006.

[68] Łukasz Maziarka, Tomasz Danel, Sławomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanisław Jastrzębski. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*, 2020.

[69] David L Mobley. Experimental and calculated small molecule hydration free energies. 2013.

[70] John S Delaney. Esol: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences*, 44(3):1000–1005, 2004.

[71] Hiroshi Sakiyama, Motohisa Fukuda, and Takashi Okuno. Prediction of blood-brain barrier penetration (bbbp) based on molecular descriptors of the free-form and in-blood-form datasets. *Molecules*, 26(24):7428, 2021.

[72] Sabina Podlewska and Rafał Kafel. Metstabon–online platform for metabolic stability predictions. *International journal of molecular sciences*, 19(4):1040, 2018.

[73] Ignacio Pérez-Correa, Pablo D Giunta, Fernando J Mariño, and Javier A Francesconi. Transformer-based representation of organic molecules for potential modeling of physicochemical properties. *Journal of Chemical Information and Modeling*, 63(24):7676–7688, 2023.

[74] Adam C Mater and Michelle L Coote. Deep learning in chemistry. *Journal of chemical information and modeling*, 59(6):2545–2559, 2019.

[75] Tânia F. G. G. Cova and Alberto A. C. C. Pais. Deep learning for deep chemistry: Optimizing the prediction of chemical patterns. *Frontiers in chemistry*, 7:809, 2019. ISSN 2296-2646. doi: 10.3389/fchem.2019.00809.

[76] Bharath Ramsundar. *Molecular machine learning with DeepChem*. PhD thesis, Stanford University, 2018.

[77] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[78] Hyun Woo Kim et al. Npclassifier: A deep neural network-based structural classification tool for natural products, 2020. URL `https://doi.org/10.26434/chemrxiv.12885494`.

[79] Kai Dührkop et al. Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. *Nature Biotechnology*, pages 1–10, November 2020. ISSN 1546-1696. doi: 10.1038/s41587-020-0740-8.

[80] Claudio N. Cavasotto and Valeria Scardino. Machine Learning Toxicity Prediction: Latest Advances by Toxicity End Point. *ACS Omega*, 7(51):47536–47546, December 2022. doi: 10.1021/acsomega.2c05693. Publisher: American Chemical Society.

[81] Hongbin Yang, Lixia Sun, Weihua Li, Guixia Liu, and Yun Tang. In Silico Prediction of Chemical Toxicity for Drug Design Using Machine Learning Methods and Structural Alerts. *Frontiers in Chemistry*, 6, 2018. ISSN 2296-2646. URL `https://www.frontiersin.org/articles/10.3389/fchem.2018.00030`.

[82] Ruili Huang, Menghang Xia, Dac-Trung Nguyen, Tongan Zhao, Srilatha Sakamuru, Jinghua Zhao, Sampada A. Shahane, Anna Rossoshek, and Anton Simeonov. Tox21Challenge to Build Predictive Models of Nuclear Receptor and Stress Response Pathways as Mediated by Exposure to Environmental Chemicals and Drugs. *Frontiers in Environmental Science*, 3, 2016. ISSN 2296-665X. URL `https://www.frontiersin.org/articles/10.3389/fenvs.2015.00085`.

[83] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. DeepTox: Toxicity Prediction using Deep Learning. *Frontiers in Environmental Science*, 3, 2016. ISSN 2296-665X. URL `https://www.frontiersin.org/articles/10.3389/fenvs.2015.00080`.

[84] Aashish Jain and Daisuke Kihara. Nntox: gene ontology-based protein toxicity prediction using neural network. *Scientific reports*, 9(1):17923, 2019.

[85] Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics*, 34(13):i52–i60, 2018.

[86] Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Opa2vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction. *Bioinformatics*, 35(12):2133–2140, 2019.

[87] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[88] Rudolf Kruse and Detlef Nauck. Neuro-fuzzy systems. In *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, pages 230–259. Springer, 1998.

[89] Dong Zhang, Xiao-Li Bai, and Kai-Yuan Cai. Extended neuro-fuzzy models of multilayer perceptrons. *Fuzzy sets and systems*, 142(2):221–242, 2004.

[90] Martin Glauer, Robert West, Susan Michie, and Janna Hastings. Esc-rules: Explainable, semantically constrained rule sets. *arXiv preprint arXiv:2208.12523*, 2022.

[91] Janna Hastings, Martin Glauer, Robert West, James Thomas, Alison J Wright, and Susan Michie. Predicting outcomes of smoking cessation interventions in novel scenarios using ontology-informed, interpretable machine learning. *Wellcome Open Research*, 8(503):503, 2023.

[92] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

[93] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, pages 2462–2467. IJCAI-INT JOINT CONF ARTIF INTELL, 2007.

[94] Luciano Serafini and Artur d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

[95] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[96] Eleonora Giunchiglia, Mihaela Cătălina Stoian, Salman Khan, Fabio Cuzzolin, and Thomas Lukasiewicz. Road-r: The autonomous driving dataset with logical requirements. *Machine Learning*, pages 1–31, 2023.

[97] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

[98] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.

[99] Emmanuel Boutet, Damien Lieberherr, Michael Tognolli, Michel Schneider, and Amos Bairoch. Uniprotkb/swiss-prot: the manually annotated section of the uniprot knowledgebase. In *Plant bioinformatics: methods and protocols*, pages 89–112. Springer, 2007.

[100] Maxat Kulmanov and Robert Hoehndorf. Deepgoplus: improved protein function prediction from sequence. *Bioinformatics*, 36(2):422–429, 2020.

[101] Maxat Kulmanov and Robert Hoehndorf. Deepgozero: improving protein function prediction from sequence and zero-shot learning based on ontology axioms. *Bioinformatics*, 38(Supplement_1):i238–i245, 2022.

[102] Maxat Kulmanov, Wang Liu-Wei, Yuan Yan, and Robert Hoehndorf. El embeddings: Geometric construction of models for the description logic el++. *arXiv preprint arXiv:1902.10499*, 2019.

[103] Janna Hastings, Martin Glauer, Adel Memariani, Fabian Neuhaus, and Till Mossakowski. Learning chemistry: exploring the suitability of machine learning for the task of structure-based chemical ontology classification. *Journal of Cheminformatics*, 13(1):1–20, March 2021. ISSN 1758-2946. doi: 10.1186/s13321-021-00500-8. URL `https://doi.org/10.1186/s13321-021-00500-8`.

[104] Martin Glauer, Adel Memariani, Fabian Neuhaus, Till Mossakowski, and Janna Hastings. Interpretable ontology extension in chemistry. *Semantic Web*, (Preprint):1–22, 2022.

[105] Adel Memariani, Martin Glauer, Fabian Neuhaus, Till Mossakowski, and Janna Hastings. Automated and explainable ontology extension based on deep learning: A case study in the chemical domain. In Roberto Confalonieri, Oliver Kutz, and Diego Calvanese, editors, *Proceedings of the Workshop on Data meets Applied Ontologies in Explainable AI (DAO-XAI 2021) part of Bratislava Knowledge September (BAKS 2021), Bratislava, Slovakia, September 18th to 19th, 2021*, volume 2998 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL `http://ceur-ws.org/Vol-2998/paper1.pdf`.

[106] Martin Glauer, F Neuhaus, T Mossakowski, Adel Memariani, Janna Hastings, P Hitzler, MK Sarker, and A Eberhart. Neuro-symbolic semantic learning for chemistry. *Compendium of Neurosymbolic Artificial Intelligence. Frontiers in Artificial Intelligence and Applications*, 369:460–484, 2023.

[107] Philippe Schwaller, Theophile Gaudin, David Lanyi, Costas Bekas, and Teodoro Laino. "found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical science*, 9(28):6091–6098, 2018.

[108] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1006. URL `https://aclanthology.org/K16-1006`.

[109] Verna Dankers, Elia Bruni, and Dieuwke Hupkes. The paradox of the compositionality of natural language: A neural machine translation case study. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.286. URL `https://aclanthology.org/2022.acl-long.286`.

[110] 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.

[111] Carlos N. Silla and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowl-*

edge Discovery, 22(1-2):31–72, January 2011. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-010-0175-9. URL http://link.springer.com/10.1007/s10618-010-0175-9.

[112] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[113] Stephen R. Heller, Alan McNaught, Igor Pletnev, Stephen Stein, and Dmitrii Tchekhovskoi. InChI, the IUPAC International Chemical Identifier. *Journal of Cheminformatics*, 7(1):23, May 2015. ISSN 1758-2946. doi: 10.1186/s13321-015-0068-4. URL https://doi.org/10.1186/s13321-015-0068-4.

[114] Paul J Karol. The inchi code, 2018.

[115] Martin Glauer, Fabian Neuhaus, Simon Flügel, Marie Wosny, Till Mossakowski, Adel Memariani, Johannes Schwerdt, and Janna Hastings. Chebifier: Automating semantic classification in chebi to accelerate data-driven discovery. *Digital Discovery*, 2024.

[116] Seyone Chithrananda, Gabe Grand, and Bharath Ramsundar. Chemberta: Large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

[117] Jesse Vig, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. BERTology Meets Biology: Interpreting Attention in Protein Language Models. *arXiv:2006.15222 [cs, q-bio]*, March 2021. URL http://arxiv.org/abs/2006.15222.

[118] Pooya Moradi, Nishant Kambhatla, and Anoop Sarkar. Interrogating the explanatory power of attention in neural machine translation. *arXiv preprint arXiv:1910.00139*, 2019.

[119] Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C Lipton. Learning to deceive with attention-based explanations. *arXiv preprint arXiv:1909.07913*, 2019.

[120] Sofia Serrano and Noah A Smith. Is attention interpretable? *arXiv preprint arXiv:1906.03731*, 2019.

[121] Jesse Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019.

[122] João D Ferreira, Janna Hastings, and Francisco M Couto. Exploiting disjointness axioms to improve semantic similarity measures. *Bioinformatics*, 29(21):2781–2787, 2013. Publisher: Oxford University Press.

[123] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[124] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach 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, 2019.

[125] 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.

[126] William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL `https://github.com/Lightning-AI/lightning`.

[127] Fernando Zhapa-Camacho, Maxat Kulmanov, and Robert Hoehndorf. mowl: Python library for machine learning with biomedical ontologies. *Bioinformatics*, 39(1):btac811, 2023.

[128] Martin Larralde. Developing python and rust libraries to improve the ontology ecosystem. *F1000Research*, 8:1500, 2019. URL `https://doi.org/10.7490/f1000research.1117405.1`. poster.

[129] Phillip Lord and Jennifer D Warrender. Horned-owl: Building ontologies at big data scale. In *ICBO*, pages 134–136, 2021.

[130] Till Mossakowski, Christian Maeder, and Klaus Lüttich. The heterogeneous tool set, hets. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 519–522. Springer, 2007.

[131] Parit Bansal, Anne Morgat, Kristian B. Axelsen, Venkatesh Muthukrishnan, Elisabeth Coudert, Lucila Aimo, Nevila Hyka-Nouspikel, Elisabeth Gasteiger, Arnaud Kerhornou, Teresa Batista Neto, Monica Pozzato, Marie-Claude Blatter, Alex Ignatchenko, Nicole Redaschi, and Alan Bridge. Rhea, the reaction knowledgebase in 2022. *Nucleic Acids Research*, 50(D1):D693–D700, January 2022. ISSN 1362-4962. doi: 10.1093/nar/gkab1016.

[132] Janna Hastings. *AI for Scientific Discovery*. CRC Press, June 2023. ISBN 978-1-00-088516-3. Google-Books-ID: hOS1EAAAQBAJ.

[133] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Michal Walczak, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. ISSN 1558-2191. doi: 10.1109/TKDE.2021.3079836.

[134] Martin Glauer, Fabian Neuhaus, Till Mossakowski, and Janna Hastings. Ontology pre-training for poison prediction. In Dietmar Seipel and Alexander Steen, editors, *KI 2023: Advances in Artificial Intelligence*, volume 46 of *Lecture Notes in Artificial Intelligence*, pages 31–45, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-42608-7. doi: 10.48550/arXiv.2301.08577. URL `https://doi.org/10.48550/arXiv.2301.08577`. Best paper award.

[135] Janna Hastings. Primer on Ontologies. In Christophe Dessimoz and Nives Škunca, editors, *The Gene Ontology Handbook*, volume 1446 of *Methods in Molecular Biology*, pages 3–13. Springer New York, New York, NY, 2017. ISBN 978-1-4939-3741-7 978-1-4939-3743-1. doi: 10.1007/978-1-4939-3743-1_1. URL `http://link.springer.com/10.1007/978-1-4939-3743-1_1`. Series Title: Methods in Molecular Biology.

[136] Gabriel Idakwo, Sundar Thangapandian, Joseph Luttrell, Yan Li, Nan Wang, Zhaoxian Zhou, Huixiao Hong, Bei Yang, Chaoyang Zhang, and Ping Gong. Structure–activity relationship-based chemical classification of highly imbalanced tox21 datasets. *Journal of cheminformatics*, 12(1):1–19, 2020.

[137] Frank Van Harmelen and Annette Ten Teije. A boxology of design patterns for hybrid learning and reasoning systems. *Journal of Web Engineering*, 18(1-3):97–123, 2019.

[138] Jiarui Chen, Yain-Whar Si, Chon-Wai Un, and Shirley W. I. Siu. Chemical toxicity prediction based on semi-supervised learning and graph convolutional neural network. *Journal of Cheminformatics*, 13(1):93, November 2021. ISSN 1758-2946. doi: 10.1186/s13321-021-00570-8.

[139] Wenhui Cui, Yanlin Liu, Yuxing Li, Menghao Guo, Yiming Li, Xiuli Li, Tianle Wang, Xiangzhu Zeng, and Chuyang Ye. Semi-supervised brain lesion segmentation with an adapted mean teacher model. In *Information Processing in Medical Imaging: 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings 26*, pages 554–565. Springer, 2019.

[140] Simon Flügel, Martin Glauer, Till Mossakowski, and Fabian Neuhaus. A semantic loss for ontology classification, 2024.

[141] Susan A Gelman. Learning from others: Children's construction of concepts. *Annual review of psychology*, 60:115–140, 2009.

[142] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5502–5511. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/xu18h.html`.

[143] Martin Grohe. The descriptive complexity of graph neural networks, 2023.