Processing Information about Biomolecules with Text Mining and Machine Learning Approaches

INAUGURALDISSERTATION

zur Erlangung des Doktorgrades
der Fakultät für Chemie und Pharmazie
der Albert-Ludwigs-Universität Freiburg im Breisgau

vorgelegt von

Kersten Döring
aus Staaken, jetzt Berlin-Spandau
Oktober 2015

Vorsitzender des Promotionsausschusses:	Prof. Dr. Stefan Weber
Referent:	Jun.-Prof. Dr. Stefan Günther
Korreferent:	Prof. Dr. Paul Wrede
Datum der mündlichen Kollegialprüfung:	06.11.2015

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Arbeit allein und ausschließlich unter Nutzung der direkt oder sinngemäß gekennzeichneten Zitate geschrieben habe. Weiterhin versichere ich, dass diese Arbeit in keinem anderen Prüfungsverfahren eingereicht wurde.

Kersten Döring
Freiburg, Oktober 2015

Zusammenfassung

Text Mining umfasst eine Vielzahl von Methoden zur Extraktion von Informationen aus Sammlungen von Texten. Der größte frei zugängliche Suchdienst für biomedizinische Forschungsergebnisse ist *PubMed*. Die Programmbibliothek *PubMed2Go* wurde im Rahmen dieser Arbeit entwickelt um beliebige Datensätze der *PubMed* zu indexieren und durchsuchbar zu machen. Desweiteren ermöglicht die Verwendung eines speziellen Dateiformats die einheitliche Nutzung weiterführender Anwendungen zur Wissensgewinnung. Mit Hilfe der geeigneten Infrastruktur konnten Modelle zur Vorhersage von funktionellen Beziehungen zwischen chemischen Kleinstrukturen und Proteinen in Sätzen von Texten der *PubMed* entwickelt werden. Der gewählte Ansatz des maschinellen Lernens wurde auf Grundlage eines eigens neu annotierten Datensatzes evaluiert. Die erfolgreiche Verwendung einer *Support Vector Machine* mit zwei verschiedenen *Kernel*-Funktionen resultierte in einem kreuzvalidierten F-Maß von ca. 80 %. *Text Mining* ermöglicht die effiziente Verknüpfung von Informationen aus der Literatur mit strukturellen oder sequenzbasierten Information aus spezifischen Datenbanken. Eine Plattform, die auf diese Weise miteinander verbundene Quellen über Annotationen von Experten bereitstellt, ist *ChemIDplus*. Die Toxizität darin enthaltener chemischer Kleinstrukturen wurde hinsichtlich der mittleren letalen Dosis auf Grundlage von molekularen Deskriptoren mit einer kreuzvalidierten Genauigkeit von bis zu 91 % vorhersagt. Die vier verwendeten algorithmischen Klassifierungsansätze lauten Entscheidungsbaum, *Random Forest*, künstliches neuronales Netz und *Support Vector Machine*. Der *Random Forest* erzielte das beste Ergebnis mit einem *Area Under Curve*-Wert von ca. 97 %. Das Synchronisieren annotierter Daten ist eine komplexe Aufgabe, die in dieser Arbeit im Zusammenhang mit der *StreptomeDB* beschrieben wird, einer Datenbank basierend auf verschiedenen Informationsquellen der Bakteriengattung *Streptomyces*. Es wurde ein *Workflow* zur Aktualisierung der Datenbank entwickelt, mit Hilfe dessen ca. 1600 neue Strukturen von ca. 600 verschiedenen Organismen in *StreptomeDB* integriert werden konnten, inklusive einer Fülle von Synthesewegen und Aktivitäten. Die hier vorgestellten Ergebnisse zeigen, dass die Nutzung von maschinellem Lernen in Kombination mit automatisiertem *Text Mining* und manueller Annotation zur Neuverknüpfung und Generation von Wissen führt.

Abstract

Text Mining approaches cover a range of methods to extract information from usually unstructured literature resources. The largest freely available repository to search for this information is PubMed. Within the amount of different software solutions to gain knowledge from texts, the newly developed software library PubMed2Go provides a unified approach to index abstracts and their meta information on a local computer, thus making them easily searchable for the user. With an appropriate infrastructure, sophisticated approaches like machine learning can be used to learn and predict patterns in texts. Such models were built within this thesis for extracting functional compound-protein relationships from sentences of PubMed abstracts, applying two different kernels with support vector machines. The approach reached an F_1 score of around 80 %, based on a newly developed and annotated benchmark data set. Text mining enables the efficient connection of textual information to other sources, like structures of chemical substances and sequences of proteins, by mapping their synonyms in texts to unique identifiers in specific databases. ChemIDplus is such an information repository including expert annotations. It was used for toxicity prediction of small molecules based on their median lethal dose. The machine learning classifiers decision tree, random forest, artificial neural network, and support vector machine reached an accuracy of up to 91 % with different sets of molecular descriptors. The best result was achieved by the random forest approach with an area under curve value of around 97 % on a clearly separated data set. The synchronisation of user-annotated data with information sources like textual and structural identifiers is a complex task, described in this thesis for the StreptomeDB, a database containing different information about the bacteria genus *Streptomyces*. Around 1,600 structures were included to the new database version via the presented update pipeline, produced by around 600 *Streptomyces* strains and containing a range of curated synthesis pathways as well as activities. The presented results prove that a combination of machine learning with automated text mining and manual curation is a valuable approach, leading to linkage of published information and generation of new knowledge.

Contents

List of Figures

List of Tables

1. Introduction

Publications are the most relevant information source for scientists to communicate their research findings. PubMed[1] is the largest open biomedical literature repository. It can be used for information retrieval, the technical term marking the process of searching for specific articles, and for the application of text mining methods. Such methods aim at information extraction from unstructured texts. This involves restructuring textual information, e.g. in relational databases, automatised text annotations with entities, such as chemical substances, proteins, diseases, etc., and conclusions about relations in the text. The process of annotating entities in texts is referred to as named entity recognition. Besides other approaches of analysing texts, it enables the possibility of counting how often different entities co-occur. This concept was applied in the Web services Compounds In Literature[2] (CIL) [1] and protein-literature investigation for interacting compounds[3] (prolific) [2].

The publication of CIL can be considered as the initiation for the work on two of the herein presented projects. The Web service CIL automatically finds names, structures, and similar structures with their co-occurring proteins in PubMed articles. The names of the chemical structures are normalised to PubChem Compound IDs [3] and the gene or protein synonyms are assigned to UniProt IDs [4]. The most similar chemical substances for a given compound are found by comparing their internally stored fingerprints[4] with the Tanimoto coefficient[5]. The basic idea of not only searching for a molecule, but also for all similar structures, to include molecules with potentially the same or similar function, is called structure-activity relationship (SAR) [5]. Prolific is similar to CIL, but it searches for protein and gene names with all co-occurrences of chemical substances. Most similar proteins are provided with the Basic Local Alignment Search Tool (BLAST) search [6] and mapped to UniProt

[1] `http://www.ncbi.nlm.nih.gov/pubmed`
[2] `http://www.pharmaceutical-bioinformatics.de/cil`
[3] `http://www.pharmaceutical-bioinformatics.de/prolific`
[4] `http://openbabel.org/docs/dev/Features/Fingerprints.html`
[5] `http://www.daylight.com/dayhtml/doc/theory/theory.finger.html`

gene symbols. All co-occurrences in CIL and prolific are shown in a heat map as illustrated in Figure 1.1.

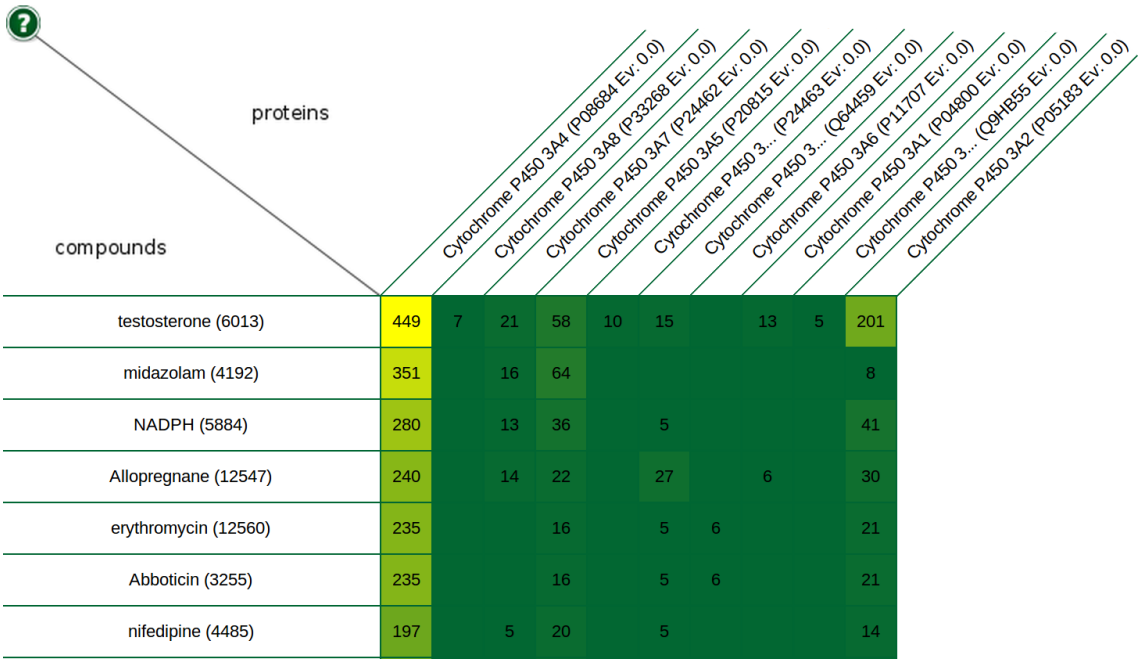


Figure 1.1.: Prolific heat map. The query protein Cytochrome P450 3A4 (CYP3A4) and all co-occurring chemical substances are shown in the first column. The most similar proteins are other CYPs. The colours illustrate the number of co-occurrences per abstract of each compound-protein pair. While the compounds with high frequencies will be metabolised by CP3A4 with a high probability, this cannot be automatically implied for the other CYPs.

CIL and prolfic are based on a PostgreSQL relational database schema[6]. The named entity recognition of PubChem compounds stored in this database was performed by searching a Xapian full text index[7] and the protein annotation was performed by the Web service Whatizit [7]. Large progress has been made in the biomedical domain of text mining and natural language processing [8]. This includes the identification of protein-protein interactions [9], compound-protein interactions [2, 10], drug-drug interactions [11, 12], and the overall connection to diseases [13]. Nevertheless, research efforts are still hindered by a lack of standardised ways to process the vast amount of data. This matter can be divided into two subjects. First, there is the problem of interoperability between different natural language processing tools for named entity recognition and relation extraction. Second, literature-related data needs to

[6] `http://www.postgresql.org`
[7] `http://xapian.org`

be prepared for large-scale applications. Both topics have been approached with the first project described in this thesis, named PubMed2Go. The software library builds a PostgreSQL relational database and a Xapian full text index on PubMed citations using Python. It can be applied either to the complete PubMed data set or an arbitrary subset of downloaded PubMed XML files. The issue of using different natural language processing tools with a literature repository like PubMed without much programming effort is approached by the implementation of an interface to a standardised interchange format, called BioC. Therefore, PubMed2Go presents an ideal starting point for the development of more sophisticated text mining methods as described in Section 3.1.

The linkage of PubMed, PubChem, and UniProt in CIL and prolific offers several possibilities to connect text mining, cheminformatics, and bioinformatics. Genes and proteins can be grouped by their molecular function as annotated in Gene Ontology[8] (GO) [14, 15] and chemical substances can be investigated in terms of their druglikeness [16]. If a chemical substance and a protein are co-occurring with a high frequency, they will probably have a functional relationship, although the type is not known [17]. If a compound occurs frequently together with a protein and a similar compound does not, this molecule might be interesting as well. This statement can be refined by restricting the co-occurrence principle to sentences and furthermore, to enclose a relationship verb [2]. Nevertheless, co-occurrences are not reliable in the case of rarely mentioned relationships. This fact motivated the second project described in this thesis. Machine learning models were developed to extract functional compound-protein relationships from sentences of PubMed abstracts as described in Section 3.2. In theory, such a classifier can be imagined similarly to a simplified human brain, parsing the structure of given information and learning to put emphasis on specific words and their contexts [18]. The implementation of compound-protein relationship models involved different natural language processing steps to restructure considered sentences into model-specific formats. Furthermore, two data sets were curated to train the machine learning models, with and without enclosed relationship verbs. The selected methods for the applied types of machine learning models were already benchmarked by Tikk *et al.* on different text corpora for the extraction of protein-protein interactions. It will be shown that these models can be particular useful in the process of searching for compound-protein relationships in texts as well (Section 4.2).

[8] `http://geneontology.org`

The machine learning approaches applied in this thesis are not only related to texts, but also to the research area of toxicology as described within the third project in Section 3.3. Toxicity predictions were performed which deal with the amount of a substance that has to be consumed to be potentially lethal. These predictions are made with a data set containing annotated levels of toxicity, based on intravenous application in mice. (Section 2.3.3). That the dose makes the poison, was already stated by Paracelsus (1493 to 1541), but the question why a substance is actually toxic can be particularly difficult to answer due to the complex issues of pharmacokinetics and pharmacodynamics [5]. Pharmacodynamics are referred to as what the chemical does to the body [5]. These effects can be modelled with methods from molecular biology and biochemistry [19, 20]. Pharmacokinetics, describing what the body does to the chemical, deals with the question how the molecule reaches the target tissue and in which concentration. In physiologically based pharmacokinetic models, this question is approached by solving the issues of absorption, distribution, biotransformation, and elimination (ADME) mathematically [5]. Considering the process of drug development, many candidate substances fail because of a lack of drug efficacy and presence of dose-limiting toxicity [5, 21]. Therefore, improving *in silico* toxicology models will save time, money, and animal lives [20, 22, 23, 24]. While drug development issues are related to the United States Food and Drug Administration (FDA) agency, the research area of toxicology in general is important for the United States Environmental Protection Agency (EPA) as well [5]. Using machine learning models on a simplified distinction of toxicity classes to find biomolecular patterns for the prediction of toxicity has led to substantially good results within this thesis, but leaves the interpretation of identified descriptors open. Part of this topic is discussed in Section 4.3.

Machine learning can be divided into supervised learning and unsupervised learning [25]. The latter approach is able to group given examples with a similar pattern into the same class, without knowing their class assignment. This can be done with clustering methods or principal component analysis. In the case of supervised learning, the algorithm is taught which training examples belong to which class. Subsequently, the model is able to predict unknown examples by applying the extracted pattern it had learned in the training period. A pattern is a group of mathematically encoded features which might not be directly clear for the user by visual inspection of the data set. If parents teach their child the difference between a cup and a glass, they will probably use features like material, colour, shape, or the presence of a

handle. In case of the algorithm, these features can be entered into a table with the class labels cup or glass and different examples given. Nominal features like colours can also be encoded with numbers.

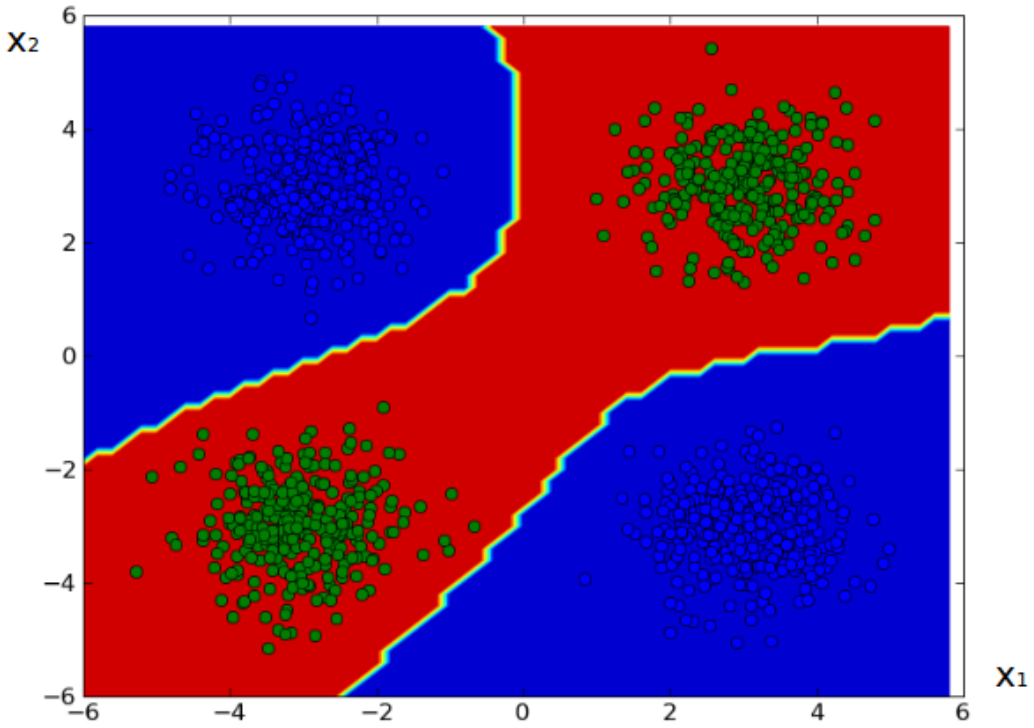


Figure 1.2.: XOR problem [20]. In the XOR problem, one of the two classes is defined by the criterion $x_1 == x_2$. This class is illustrated with green dots on red background. The other class ($x_1 \neq x_2$) is displayed with blue dots on blue background. The artificial neural network classifier used in this case learned to distinguish between green dots and blue dots as training examples. The red background colour displays the area in which the model still classifies a pixel as belonging to the group of green dots. Everything shown in blue will be predicted as the other class. The scatter plots were generated as two-dimensional Gaussian distributions. Due to the random initialisation, the discriminative function will lead to different results for each repetition.

Classical machine learning examples are email spam filter or face recognition in cameras. Supervised machine learning classifier examples in bioinformatics are e.g. the prediction of major histocompatibility protein I (MHC-I) binding peptides [26], cell penetrating peptides [27], and membrane proteins [28]. In these cases, features are amino acid-related descriptors, e.g. their distribution, net charge, or hydrophobicity.

A classification problem can also be displayed graphically as shown in the two-dimensional exclusive OR function (XOR problem) in Figure 1.2. This figure also represents a problem of non-linear separability, because there is no possibility to separate the green dots and the blue dots with a single linear line. High-dimensional classification problems are difficult to display, but the principle remains the same, dealing with hyperplanes and coordinates in space.

The machine learning approaches used in this thesis to separate toxicity classes are artificial neural networks, support vector machines, decision trees, and random forests. The features are physico-chemical properties, functional groups, and substructures of the molecules. In case of the prediction of functional compound-protein relationships, the two approaches in comparison are the support vector machine and the sparse regularized least squares classifier. Computed features are terms in the sentences and their part-of-speech tags. In machine learning, the quality of the model is dependent on the data set. Especially in the case of toxicity prediction, different descriptor sets are evaluated to try to find explanations for the patterns used by the classifier.

The fourth project within this thesis, named StreptomeDB, covers data which cannot yet be annotated automatically by text mining and pattern matching methods, because the reliability would not be as high as it is by involving expert knowledge. The core of StreptomeDB is based on curated *Streptomyces* strains and the chemical compounds produced by them, possibly with annotated activities and synthesis routes. The process of text curation and drawing of annotated structures that do not yet exist in PubChem is time-consuming, but of high benefit. The project was published first in 2013 [29] and is also connected to PubMed2Go. Until September 2015, StreptomeDB was already cited several times. Therefore, the data can be used with a range of different research focuses, as discussed in Section 4.4.

Streptomyces belong to the Gram-positive actinobacteria and contain a high content of guanine and cytosine DNA bases [30]. This genus is of so much interest for the scientific community, because it occupies a range of different terrestrial and aquatic habitats. Furthermore, they produce more than 60 % of all known antibiotics [31]. This includes approved drugs, such as tetracycline [32], daptomycin [33], and chloramphenicol [34]. The probably most popular drug isolated from *Streptomyces* is also the first one, named streptomycin and identified by Albert Schatz in 1943 [35]. Natural compounds from *Streptomyces* can also be produced as semi-synthetic drugs [36], as in the case of griselimycin [37]. Beside antibiotics, there are also biologically active

and therapeutically important drugs, like the anti-parasitic agent avermectin [38], the immunosuppressant rapamycin [39], and the lipase inhibitor lipstatin [40]. There is still a range of possibilities to discover new drugs from *Streptomyces* [41]. Considering this excerpt of drugs related to *Streptomyces*, there is the requirement of an update pipeline supporting the process of computer-assisted curation of new publications to keep the centralised knowledge about these versatile bacteria up-to-date. The steps involved in this workflow are described in Section 3.4. There are several new features to further explore the chemical diversity of compounds and the phylogeny of *Streptomyces* strains in the new update version of StreptomeDB, but these functions will be presented in detail in the follow-up publication [42].

While the next chapter illustrates methods used in one or several of the results' sections, Section 3.1 to Section 3.4 describe the projects PubMed2Go, functional compound-protein relationship prediction, toxicity prediction, and StreptomeDB in an enclosed way. Thereby introduced related work and future prospects are discussed in the last chapter.

2. Methods

2.1. Text Mining-related Methods

2.1.1. Extensible Markup Language

The Extensible Markup Language (XML) is a commonly used text-based data exchange format. It is used to structure text fragments with key-value pairs, describing a kind of categorisation for each block it is belonging to. These categorisation or structure types are defined in a document type definition (DTD) file such as the PubMed DTDs[1]. The documents that are freely available via the PubMed search engine are mostly database entries from *Med*ical Literature Analysis and Retrieval System On*line* (MEDLINE). Within this thesis, they will be referred to as PubMed XML files. The complete XML data set of PubMed can be downloaded from the NLM FTP server, including example data sets[2,3]. An example for XML elements in PubMed articles enclosing blocks of text is shown in Figure 2.1. These key-value elements can be processed with an XML parser. The Python[4] parser used in the PubMed2Go project (Section 3.1) uses the ElementTree XML application programming interface[5] (API) which is similar to a Simple API for XML (SAX) interface[6].

A more specific format is described with the BioC XML format for biomedical annotations (Section 3.1.3). In general, the text fragments themselves are still unstructured, but they can be selected by their categorisation type. The reason not to use Office documents for automated text processing tasks is that these files are

[1] `http://www.nlm.nih.gov/databases/dtd`
[2] `http://www.nlm.nih.gov/databases/journal.html`
[3] `ftp://ftp.nlm.nih.gov/nlmdata/sample/medline`
[4] `https://www.python.org`
[5] `https://docs.python.org/2/library/xml.etree.elementtree.html`
[6] `http://stackoverflow.com/questions/192907/xml-parsing-elementtree-vs-sax-and-dom`

```
−<MedlineCitationSet>
  −<MedlineCitation Owner="PIP" Status="MEDLINE">
     <PMID Version="1">12255379</PMID>
     [...]
    −<Journal>
       <ISSN IssnType="Print">0002-9955</ISSN>
      −<JournalIssue CitedMedium="Print">
         <Volume>159</Volume>
         <Issue>3</Issue>
        −<PubDate>
           <Year>1955</Year>
           <Month>Sep</Month>
           <Day>17</Day>
         </PubDate>
       </JournalIssue>
       <Title>Journal of the American Medical Association</Title>
       <ISOAbbreviation>J Am Med Assoc</ISOAbbreviation>
     </Journal>
     <ArticleTitle>Association of maternal and fetal factors [...]</ArticleTitle>
     <AuthorList CompleteYN="Y"> [...] </AuthorList>
     [...]
    −<MeshHeadingList>
      −<MeshHeading>
         <DescriptorName MajorTopicYN="N" UI="D001519">Behavior</DescriptorName>
       </MeshHeading>
       [...]
     </MeshHeadingList>
     <AbstractText>Pregnancy, delivery, and neonatal records of [...]</AbstractText>
    −<KeywordList Owner="PIP">
       <Keyword MajorTopicYN="N">Behavior</Keyword>
       [...]
     </KeywordList>
   </MedlineCitation>
 </MedlineCitationSet>
```

Figure 2.1.: PubMed XML format. The whole example data set is enclosed with a MedlineCitationSet XML tag. Every article contains an opening and closing MedlineCitation tag. The PubMed ID is represented with the key PMID and the value 12255379. Other XML elements are shown, e.g. publication date, journal, title, and text.

Table 2.1.: Command to download of PubMed XML files. The tool wget can be used in the command-line of Linux systems to download files, e.g. from File Transfer Protocol (FTP) servers. In this case, the NCBI EFetch interface is used. The output file medline_00000000.xml will contain the XML documents with the PubMed IDs 25006566 and 25005174.

Command	
wget	-O medline_00000000.xml "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed& id=25006566,25005174& retmode=xml"

encoded in a binary format which cannot be read directly by programming languages like Python or Java[7]. The parsing step (reading XML documents) can be used for storing the sub-categorised text fragments in a relational database as described in the next subsection.

The XML files can be downloaded via the NCBI interface EFetch[8]. An example as applied with PubMed2Go[9] is shown in Table 2.1.

2.1.2. Structured Query Language

Structured Query Language (SQL) commands and tables are used in all projects described in this thesis except the toxicity prediction. Considering the XML example in the last subsection, every key-value pair of XML elements can be processed and stored in a relational database, e.g. by using PostgreSQL. Therefore, the data can be queried with SQL commands instead of parsing the text file again. Different approaches exist, such as building SQL statements directly in an SQL script or using a programming language like Python. Text elements can be processed with Python and the module Psycopg[10] can be used to execute an SQL query. PubMed2Go uses a different approach in which the SQL tables and their dependencies are implemented as Python classes (Section 3.1). This approach is called object-relational mapping. It is implemented in the Python module SQLAlchemy[11]. With this software, elements can be pushed to SQL tables with Python functions that

[7] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
[8] `http://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4.EFetch`
[9] `https://github.com/KerstenDoering/PubMed2Go/tree/master/data`
[10] `http://initd.org/psycopg`
[11] `http://www.sqlalchemy.org`

implicitly use SQL statements in Psycopg. An SQL query example is shown in Table 2.2 and its result in Table 2.3. This SQL statement can also be executed with the PostgreSQL Open Source administration and development platform PGAdmin (Figure 2.2).

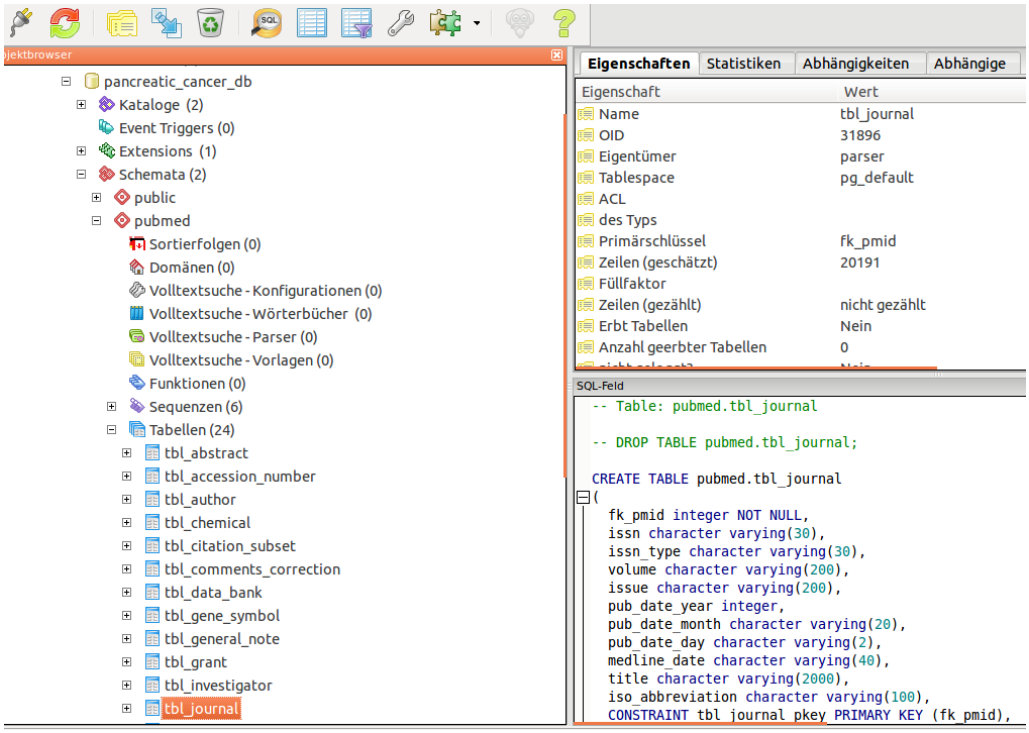


Figure 2.2.: PosgreSQL queries in PGAdmin. The PGAdmin platform can be configured to connect to a database in the network, e.g. pancreatic_cancer_db (Section 3.1). A part of the different tables in the schema pubmed is shown on the left side. The SQL code to generate the selected table is shown on the right side. The magnifying glass symbol can be clicked to run queries like the one shown in Table 2.2.

Another approach of storing textual data is to use a NoSQL database like MongoDB[12], e.g. with the Python interface PyMongo[13]. The name relational databases implies that the concept of SQL tables is to use different tables which are connected with foreign keys. In the case of NoSQL tables, the usage commonly relies on one large table containing all information. One motivation to use this technology is that it can still be applied where an SQL join of different entity types and their references is too slow. An example for a NoSQL query is illustrated in Section 3.2.4. It is referring to

[12] `https://www.mongodb.org`
[13] `https://api.mongodb.org`

a case in which the MongoDB is needed too store a huge number of small molecules and proteins that were joined within the back end of the Web service prolific [2].

Table 2.2.: PostgreSQL query to select journals published before 1950. This query selects PubMed IDs, publication titles, and journal names, based on the PubMed2Go pancreatic cancer data set (Section 3.1). The information is located in two different tables. Therefore, the selection includes a join of two tables via the PubMed ID as a foreign key.

SQL command	
SELECT	mc.pmid, mc.article_title, mj.pub_date_year, mj.title
FROM	pubmed.tbl_medline_citation mc
INNER JOIN	pubmed.tbl_journal mj
ON	mj.fk_pmid = mc.pmid
AND	mj.pub_date_year < 1950
ORDER BY	mj.pub_date_year;

Table 2.3.: Results for query in Table 2.3. There are six publications from the time before 1950, related to pancreatic cancer. Pancreaticoduodenectomy is the medical term for a surgery removing the pancreas and part of the surrounding organs. Today, it is still practised similarly to the procedure successfully published first.

PubMed ID	Title	Year	Journal
17856333	PANCREATIC CANCER AND ITS TREATMENT BY IMPLANTED RADIUM.	1934	Annals of surgery
20267924	The present status of pancreaticoduodenectomy for carcinoma of the ampulla of Vater and of the head of the pancreas.	1947	The Journal of the International College of Surgeons
20259580	[Pancreatic cancer].	1947	Cincinnati journal of medicine
20242992	The diagnosis of pancreatic cancer.	1947	Medical times
18149952	Exfoliated pancreatic cancer cells in duodenal drainage; a case report.	1949	Cancer
15396889	[Radical treatment of pancreatic cancer].	1949	Gazette des hôpitaux civils et militaires de l'Empire ottoman

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
#Kersten Doering 19.08.2015

# Xapian interface in Python
import xappy

# each text fragment in this list will be indexed as a Xapian document (with ID)
strings = ["hello world", "hello user","used"]
# set Xapian full text index connection
# the directory xapian will be created by IndexerConnection()
conn = xappy.IndexerConnection("xapian")
# add a new field with the name text
conn.add_field_action('text', xappy.FieldActions.INDEX_FREETEXT, language='en')
# store the content of each document (text)
# this option is useful for displaying the matching text in case of a hit
conn.add_field_action('text', xappy.FieldActions.STORE_CONTENT)

# iterate over the text elements in the list, the new ID will be the list index
for index,string in enumerate(strings):
    # new Xapian document
    doc = xappy.UnprocessedDocument()
    # add text
    doc.fields.append(xappy.Field("text", string))
    # add ID
    doc.id = str(index)
    # add document to the Xapian full text index
    # it will be stored after the end of this for-loop
    conn.add(doc)
# push changes in the Xapian full text index to the database
conn.flush()
# close connection
conn.close()
```

Figure 2.3.: Xapian full text index. These steps are important to index text documents with Xapian, e.g. parsed text files, entries from a relational database, a list of names, etc. In case of PubMed2Go, the document IDs are PubMed IDs and the indexed fields are titles, abstract texts, MeSH terms, keywords, and chemical substances.

2.1.3. Xapian Full Text Index

In this thesis, the Xapian Python interface Xappy[14] is used. The library is easy to use, because an index can basically be built in a few steps, as provided in Figure 2.3. An appropriate search example is illustrated in Figure 2.4, with the result shown as text that was commented out. More advanced examples with conditional queries like AND or NEAR can be found in the PubMed2Go documentation[15]. One important concept of Xappy is to directly index fields and to set up documents with a customised document ID. The prefix in the search result of Figure 2.4 is an XA, which corresponds to a user-defined field X with the index specification A. If there was another field such as title in addition to the field text, the second query would

[14] https://pypi.python.org/pypi/xappy
[15] https://github.com/KerstenDoering/PubMed2Go/wiki

show an XB. The Z in the query syntax represents the search for a stemmed version of the term[16].

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
#Kersten Doering 19.08.2015

# Xapian interface in Python
import xappy

# set path to Xapian full text index database for searching
searchConn = xappy.SearchConnection("xapian")
searchConn.reopen()
# list of search terms
querystrings = ["hello","using"]
# iterate over search terms
for querystring in querystrings:
    # define query to search in the field 'text'
    q = searchConn.query_field('text',querystring)
    # debug: show query
    print "search query: ", q
    # find results
    results = searchConn.search(q, 0, searchConn.get_doccount())
    # show results with ID and the text in which the match appeared
    for r in results:
        print r.id, r.data['text'][0]
# close connection
searchConn.close()

"""
#output
search query:  Xapian::Query((ZXAhello:(pos=1) AND_MAYBE XAhello:(pos=1)))
0 hello world
1 hello user
search query:  Xapian::Query((ZXAuse:(pos=1) AND_MAYBE XAusing:(pos=1)))
2 used
"""
```

Figure 2.4.: Xapian full text search. The output of this search, based on the index built in Figure 2.3, is shown below the programme code. The Xapian documents "hello world" and "hello user" are matched exactly. The matched document "used" is resulting from the common word stem "us" in "using".

2.1.4. Entities

An entity describes a type or class of words in a text, e.g. genes and proteins, small molecules and drugs, or diseases. Different approaches exist that identify synonyms of these categories, e.g. with dictionaries or by using machine learning.

[16] `http://xapian.org/docs/omega/termprefixes.html`

2.1.4.1. Genes and Proteins

Whatizit [7] is a Web service that provides named entity recognition of genes and proteins, gene ontology terms, diseases, organisms, and chemical substances. It can be accessed via a Simple Object Access Protocol (SOAP) with programming languages or via a Web page[17]. Whatizit uses a dictionary-based approach using UniProt synonyms [4]. Therefore, it offers a fast protein-tagging service.

A more sophisticated approach is used by GeneTUKit [43]. The software uses conditional random fields (CRF) [44], a machine learning technique which considers the word neighbourhood of a candidate synonym, an Entrez Gene-based dictionary [45], and ABNER [46], another named entity recognition software. If two of these three approaches vote for a possible gene or protein, the synonym will be further processed by a ranking algorithm, considering different features like the local context and the global text, e.g. the whole abstract or full text article. Therefore, it is possible to perform gene normalisation, which means mapping an organism-specific gene ID to a synonym. Finally, a support vector machine generates a confidence score specifying a level of certainty for each gene ID prediction. Unfortunately, the GeneTUKit source code is not available, but the software can be executed as a Java Archive (JAR) application [43]. There is a prototype implementation[18] using PubMed2Go for downloading relevant articles and getting the Entrez Gene IDs with GeneTUKit. The result is a set of annotated abstracts in which the Entrez Gene IDs are mapped to UniProt IDs. In a test run with 20,000 abstracts, the average runtime for 1,000 abstracts was around 1 h. E. Abasian worked on this project during her Master Thesis under my supervision [47]. GeneTUKit and GNAT [48] are the most commonly known and best performing tools for gene normalisation. One difference is that GNAT is restricted to 20 common model organisms like human, mouse, rat, and *E. coli*.

2.1.4.2. Small Molecules and Drugs

The Web service CIL [1] processes PubChem compounds [3] applying the Hettne-Rules [49]. These rules were defined to filter synonyms which will probably result in false positive hits. This includes eight rules, e.g. the short token filter rule, excluding

[17] `http://www.ebi.ac.uk/webservices/whatizit`
[18] `https://github.com/ElhamAbbasian/GeneTUKit-Pipeline`

synonyms that result in a single letter or number after removing stop words[19], and the dosage rule, basically removing units or measurements. After the complete PubMed XML data set was searched for PubChem compounds with Xapian, a frequency table was build in which it was counted in how many abstracts each chemical substance synonym was found. PubChem synonyms are particularly useful to map named entity recognition terms to real structures, but a lot of synonyms are not useful. Considering the list of matched structure names, the first synonym was "(or)" with a frequency of 5,777,083. This phenomenon has got two reasons. Xapian does not take care for brackets and other special characters. The search engine replaces these parts with wildcards, trying to match the overall term. The second reason for such a result is that the Xapian search is case-insensitive. The synonym identified here is "(OR)" and belongs to the molecule methyl 2-[4-(4-chlorobutanoyl)phenyl]-2-methylpropanoate[20]. The whole list contained around 302,046 identified terms matched with a with a PubChem synonym list[21], both from October 2014. The first 7,000 synonyms were checked in a semi-automatic way by O. Thomas[22]. All terms following afterwards occurred with a frequency of around 1,000 and less. Considering the length of the list, structure names with less occurrences will less likely show false positive hits.

2.1.4.3. Co-occurrences and Relationship Verbs

If two biomolecules appear together in a text, they will be referred to as co-occurring. A comparably high number of such pairs of biomolecules can be use to predict a relationship, e.g. between proteins or proteins and chemical compounds (Section 3.2). This is commonly known and e.g. mentioned by Tikk *et al.* [9] or Jensen *et al.* [17]. Furthermore, Tikk *et al.* refer to the idea to refine this concept by including relationship words and verbs [50]. This was independently done in the publication of prolific [2]. A list of potentially important verbs was created and iteratively searched in all PubMed abstracts with an enclosing structure of a chemical compound and a protein. Every verb which did not seem to lead to appropriate results after inspecting a few examples was removed. After completion, verb forms in all tenses were created and saved in a PostgreSQL relational database. Therefore, the Web service prolific contains an additional search option to enclose a relationship verb [2]. Nevertheless,

[19] `http://www.ncbi.nlm.nih.gov/books/NBK3827/table/pubmedhelp.T.stopwords`
[20] `https://pubchem.ncbi.nlm.nih.gov/compound/10062338`
[21] `ftp://ftp.ncbi.nlm.nih.gov/pubchem/Compound/Extras/CID-Synonym-filtered.gz`
[22] `http://www.pharmaceutical-bioinformatics.de/main/members`

these verbs were not annotated with part-of-speech tags, such that some examples might be nouns, especially in the case of the gerund with the suffix -ing. The quality of this approach in relation to functional compound-protein relationships is analysed in Section 3.2.

2.2. Machine Learning Approaches

The methods described in this section were applied in the toxicity prediction project (Section 3.3) and the extraction of functional relationships between chemical compounds and proteins from sentences in PubMed abstracts (Section 3.2). Part of the presented figures and explanations are related to my Bachelor Thesis in 2009 [20] and L. M. Grüger's Bachelor Thesis [51], supervised by me.

2.2.1. Evaluation Parameters

A machine learning classifier needs to be evaluated by measuring its amount of correct and wrong predictions. This can be done with a confusion matrix (Table 2.4). Summarising the rows and columns in this table leads to a range of evaluation parameters, described in this subsection.

Table 2.4.: Confusion matrix. The column headers show the positive and negative predictions. The rows represent the actual class with the gold standard annotation. Therefore, the falsely positive predicted instances are positioned in the lower left part of the table (FP) and the false negatives can be found in the upper right part of the table (FN). True positives are abbreviated with TP and true negatives with TN.

Actual class / predicted class	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Predictions that are correct can be true positives or true negatives. All true negatives that were classified positive are called false positive. Therefore, positive instances that were classified incorrectly are false negatives. In case of functional relationships (Section 3.2), a positive example refers to a true relationship. Considering the toxicity prediction project (Section 3.3), the positive class is the more toxic class.

2.2.1.1. Sensitivity

Sensitivity, also called recall or TP rate, measures how many of the really true examples in the data set were identified as positive cases. Based on Table 2.4, the following formula refers to the first row.

$$R = \frac{TP}{TP + FN}$$

2.2.1.2. Specificity

The specificity measures how many of the really negative examples were recognised as such. Therefore, it can also be considered as the sensitivity of the negative class. By convention, this measurement refers to the second row in Table 2.4. In the WEKA confusion matrix (Section 2.2.8 and Section 3.3), there is only the recall of class a and the recall of class b given. In case of named entity recognition tasks, this measurement is usually not considered, because the number of negative examples refers to the rest of the terms given in a text. The FP rate is defined as $1 - S$.

$$S = \frac{TN}{TN + FP}$$

2.2.1.3. Precision

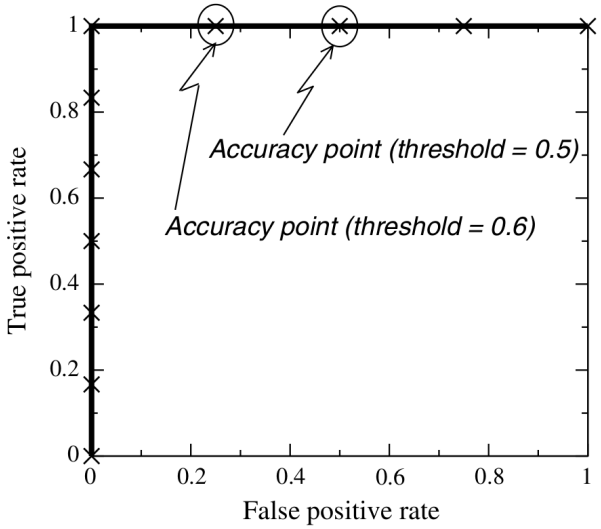
In contrast to the recall, the precision measures how many of the positive predictions are actually really positive examples. This evaluation parameter is important for text mining-related topics. The precision of the negative class is not considered, because a classifier is commonly judged by its ability to find the positive cases. The value is calculated with the entries in the first column of Table 2.4.

$$P = \frac{TP}{TP + FP}$$

2.2.1.4. $\mathbf{F_1}$ Score

This evaluation parameter can be considered as the weighted average of precision and recall. Its value ranges from zero to one.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$



no.	True	Hyp	
1	**p**	**Y**	0.99999
2	**p**	**Y**	0.99999
3	**p**	**Y**	0.99993
4	**p**	**Y**	0.99986
5	**p**	**Y**	0.99964
6	**p**	**Y**	0.99955
7	**n**	**Y**	0.68139
8	**n**	**Y**	0.50961
9	**n**	**N**	0.48880
10	**n**	**N**	0.44951

Figure 2.5.: Area under curve [52]. The table shows all probabilities for each prediction beginning with the largest score in a descending order. The column "True" shows a "p" for a really positive instance and "n" otherwise. The classifier's output for a positive prediction ("Y") is based on the decision whether a value is larger than a threshold of 0.5. This threshold is not optimal, because it results in two false positive predictions. Each cross in the ROC plot corresponds to a new threshold. The circled crosses illustrate the thresholds with suboptimal accuracies, which are 80 % and 90 % respectively. The optimal threshold value would be around 0.7, resulting in an accuracy of 100 %. Neverthelss, the AUC of this classifier is 100 %, which means that all examples were ranked in the correct order. Therefore, the ROC curve can be used to tune the classifier's relative performance by changing the classification threshold. Another approach is to calibrate the probability scores [52, 53].

2.2.1.5. Area Under Curve

The area under curve (AUC) can be illustrated with a receiver operating characteristic (ROC) curve. The descriptions presented in this subsection are based on a good introduction to ROC analysis by Tom Fawcett [52]. The calculation of an ROC curve is based on the fact that most classifiers do not only provide the decision whether a prediction is positive or negative, but also a probability or probability score. Sorting these scores descending and shifting a threshold over all output scores leads to a different confusion matrix for every new prediction example. Therefore, the TP and FP rates can be plotted for every threshold that changes the values in the confusion matrix, as shown in Figure 2.5.

The AUC corresponds to the percentage that is covered by the ROC curve. The maximum value is 100 %, but this performance is rarely achieved. The ROC values in the upper left area of the plot correspond to the best classification results with the highest TP rate and the lowest FP rate. A classifier with an ROC that relatively covers the diagonal cannot be considered better than random. Figure 2.5 also shows that the highest thresholds correspond to the lowest left region of the ROC curve, resulting in a very low FP rate, but also a low number of TPs. The lower the threshold, the higher the TP rate will be, probably also resulting in a higher FP rate. Therefore, the former approach is called conservative and the latter approach liberal. In case of toxicity prediction, a higher number of FPs might be a good choice. Although the resulting number of TN examples will be rather low, the number of unrecognised really toxic examples (FN) will also remain comparably low. In case of applying text mining filter options, the better approach can be using a conservative threshold, such that the user will not have to read through many FP examples. The AUC can be calculated by summing up the area below the ROC curve or by summing up the number of positively ranked examples that have a higher score than all negative examples, divided by the total number of comparisons [54]. This turns out to result in the same meaning. The AUC is defined as the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative example. Another way than optimising for the highest accuracy is to find the decision threshold with the best F_1 score [54]. This might shift the threshold into the lower left area of the ROC plot, which is the already mentioned interesting case for text mining-related tasks. In general, the F_1 score can be affected by class skews, because it is only based on part of the confusion matrix, while the AUC calculation covers both columns in this matrix.

2.2.1.6. Binary and Multi-class Classification

There are classifiers that are able to learn multi-class models, such as artificial neural networks and decision trees. In case of binary classifiers like support vector machines, there are implementations that train a model for each class against all other classes, also called the one-against-all method [25]. If e.g. three classes are given, as illustrated in Figure 2.6, the evaluation parameters explained in the last subsections can be calculated by explicitly building three two-class confusion matrices from the three-class matrix or by using the two-class matrices provided by the three

Predicted class →	A	B	C
A	a	b	c
B	d	e	f
C	g	h	i

Predicted class →	A	B/C
A	a	b+c
B/C	d+g	e+f+h+i

Predicted class →	B	A/C
B	e	d+f
A/C	b+h	a+c+g+i

Predicted class →	C	A/B
C	i	g+h
A/B	c+f	a+b+d +e

Figure 2.6.: Three-class confusion matrix [51]. The three classes A, B, and C can be combined in three different two-class data sets.

one-against-all models.

2.2.2. Cross-validation

In case of k-fold cross-validation, the data set is divided into k equally sized subsets. A model is trained k times with k-1 subsets, such that each of the k subsets is predicted once. Therefore, the confusion matrix can be built from all subsets to calculate the evaluation parameters on all instances from the data set. Cross-validation ensures that a possible effect of more easy or difficult examples and more similar or diverse instances is averaged [55]. Therefore, the classifier's performance and ability to generalise patterns from a data set can be evaluated. In this thesis, all experiments were performed with 10-fold cross-validation.

In the case of classifying relations in sentences, it is recommended to perform document-level cross-validation instead of instance-based cross-validation [54]. The difference is that in a sentence with $\binom{n}{2}$ binary entity pairs, these instances will remain in the same data set split, using the document-level cross-validation. The reason not to separate instances from the same sentence or document is that they cannot be considered as independent examples. Therefore, a split on an instance-based level would lead to information leakage and non-representative evaluation

results [54, 56, 57].

2.2.3. Artificial Neural Networks

The XOR problem mentioned in the introduction of this thesis is a non-linearly separable problem. The artificial neural network classifier is able to solve such problems by combining single neurons which can be referred to as perceptrons. This analogy of the algorithmic approach and real neurons in the brain was introduced by Hodgkin and Huxley in 1952 [58]. The concept of combining weighted excitatory and inhibitory input edges with a threshold function, deciding to fire or not, was mathematically introduced as the perceptron by Frank Rosenblatt in 1958 [59] and further refined by Minsky and Papert in 1969 [58]. Such a single perceptron is shown in Figure 2.7.

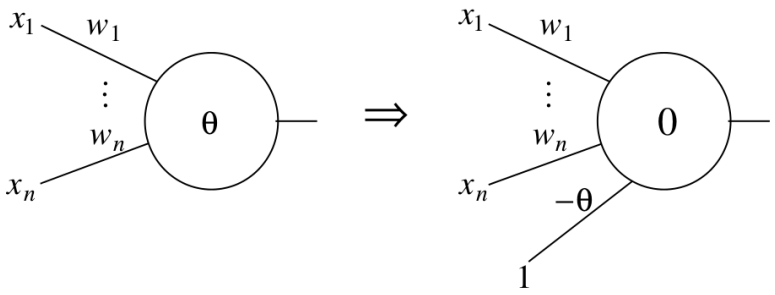


Figure 2.7.: Perceptron [58]. The left and the right perceptron can be considered as equal, because the only difference is a constant input of +1. With this bias, the right perceptron is normalised to a threshold of 0.

Mathematically, any logical function can be computed by a perceptron network [58]. The perceptron is able to solve a linearly separable problem by multiplying each input value with a certain weight and computing the sum of all products. This is equal to the vector product of an n-dimensional weight and input vector.

$$f(x) = \sum_{i=1}^{n} w_i x_i + b$$
$$f(x) = \langle w \cdot x \rangle + b$$

In the two-dimensional input space, this refers to standard linear function with the

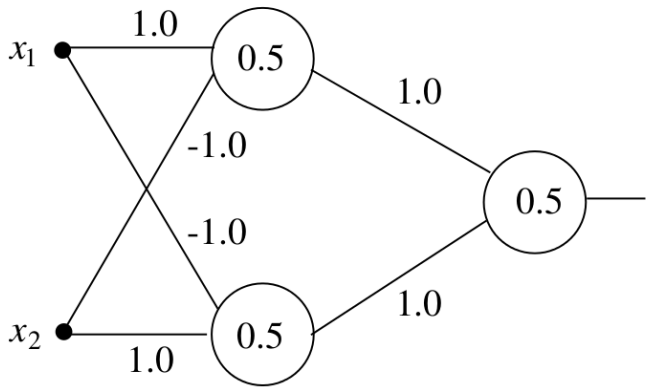


Figure 2.8.: XOR multilayer perceptron [58]. If both inputs are equal, e.g. $x_1 = 1$ and $x_1 = 1$, the input for the first layer of perceptrons is equal to 0. Therefore, the units do not fire. Subsequently, the sum of the products with 0 as the input for the last neuron is also smaller than the threshold 0.5. For unequal inputs, e.g. $x_1 = 1$ and $x_1 = 0$, the network generates the output 1.

slope a and the bias b.

$$f(x) = ax + b$$

In case of the XOR problem a multilayer perceptron with three units can be constructed in an analytical way [58], as shown in Figure 2.8.

In general, a multilayer perceptron consists of an input layer, a hidden layer and an output layer (Figure 2.9). A network can consist of multiple hidden layers.

All neurons except the input nodes contain an activation function, e.g. a sigmoid function (Figure 2.10).

The output neuron also contains an error function. It measures the squared distance of the value computed by the output function and the true label y of each of the p training instances in the data set S [58].

$$S = \{(x_1, y_1), \ldots, (x_p, y_p)\}, \;\; Y = \{0, 1\}$$
$$E = \frac{1}{2p} \cdot \sum_{i=1}^{p} (f(x_p) - y_p)^2$$

The aim is to train the network to compute the correct output function $f(x)$. This can be achieved by minimising the error function E. The network function $f(x)$ is not explicitly given, but represented as training data points in some input space.

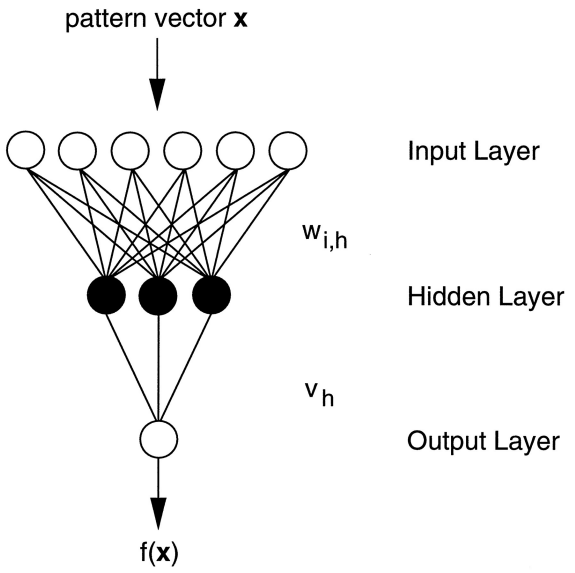


Figure 2.9.: Artificial neural network [60]. In this generalised network structure, the input dimension is $n = 6$. There are three hidden layer neurons and one output layer neuron for binary classification.

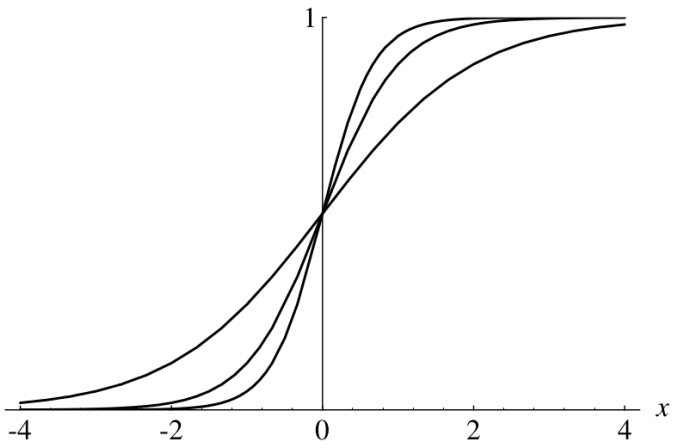


Figure 2.10.: Sigmoid function [58]. The sigmoid function $f(x) = \frac{1}{1+\exp^{-cx}}$ is shown with $c = 1$, $c = 2$, and $c = 3$. The largest parameter c refers to the curve with the steepest slope for $x = 0$.

The best approximation is computed by function composition of single neurons with sigmoid output functions. The function changes by altering the weights in every neuron layer. All weights ℓ can be changed by computing their partial derivatives to make use of the gradient descent in the error space. This is the reason for choosing a sigmoid activation function. It ensures that there are no plateau regions with gradient zero. The derivatives can be computed until a local or global minimum

is found in the error space [58]. The partial derivatives are represented with the greek symbol nabla (∇). As commonly known from two-dimensional derivatives, the derivative of the error function is set to zero.

$$\nabla E = 0$$
$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i}, \quad for \;\; i = 1, \ldots, \ell$$
$$w_t = w_{t-1} - \gamma \nabla E$$

Considering again the linear, two-dimensional space ($f(x) = ax + b$), with bias zero, this can be imagined as a quadratic function with the amount of error displayed on the y-axis and the value of the weight a on the x-axis. The slope of the tangent line (from the derivative of the quadratic function) determines the amount of change for the weight value a. This slope needs to be followed iteratively until the minimum is reached. If the slope is negative, the value of the weight grows, moving from left to right on the x-axis. If it is a positive gradient, the weight will shrink, moving from right to left on the x-axis. If the step width γ is not too large, this results in a minimum value of the error function after t steps.

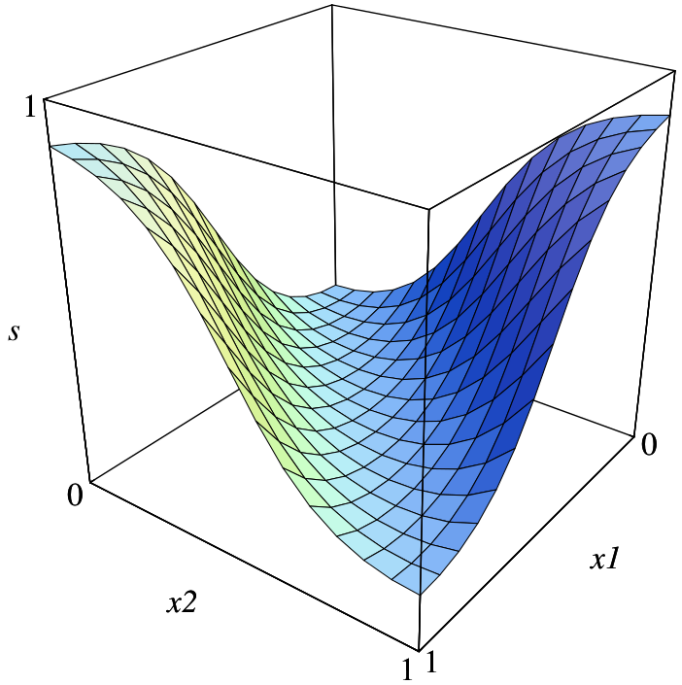


Figure 2.11.: XOR output space [58]. The value s equals the output of the network function $f(x)$. This value is computed for a range of x_1 and x_2 values from zero to one.

After the training procedure, the output function can be explicitly computed. In case of the analytical network from Figure 2.8, this results in a three dimensional plot, showing the output on the z-axis (Figure 2.11). All pairs of x_1 and x_2 generating an output above 0.5 belong to class 1 and all other points to class 0. Remembering that Figure 1.2 was computed with two-dimensional Gaussian distributions, it can be imagined that the separating line between the blue and the red area is a variation of the line of intersection with the hyperplane for $f(x) = 0.5$ in Figure 2.11.

2.2.4. Kernel Methods

While the artificial neural network makes its decision using a threshold based on approximating a probably non-linear function, approaches using kernel functions try to solve their classification problems in a linear space using the kernel trick [61]. A kernel function is used to map the input data into a (high-dimensional) feature space, in which the problem can be solved with a linear function (Figure 2.12), but the feature space is not explicitly computed. Given a trained model with a valid kernel function, the trick is that every new data point can be predicted by computing its vector product with the feature vector [62]. As well as in the case of toxicity prediction (Section 3.3), this works for the text classification task of finding functional relationships between chemical compounds and proteins (Section 3.2).

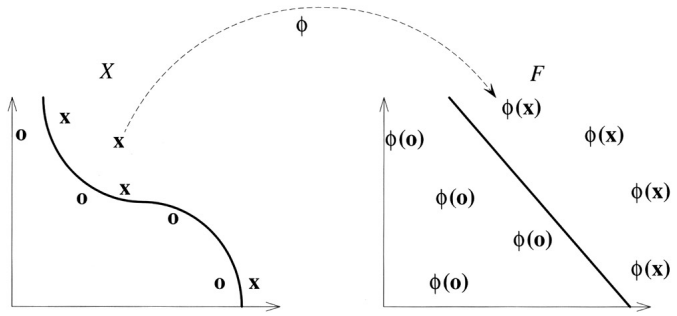


Figure 2.12.: Kernel-induced feature space [62]. Every input vector is transformed into a feature space to simplify the classification task.

2.2.4.1. Linear Support Vector Machines

In case of a linear SVM, the kernel function is just the identity function, similar to the perceptron function. Given a linearly separable problem, a function $f(x) = \langle w \cdot x \rangle + b$

needs to be trained, which classifies instances positively, if $f(x) \geq 0$. All other instances will be classified as negative examples. Figure 2.13 shows that this function can be understood as a hyperplane.

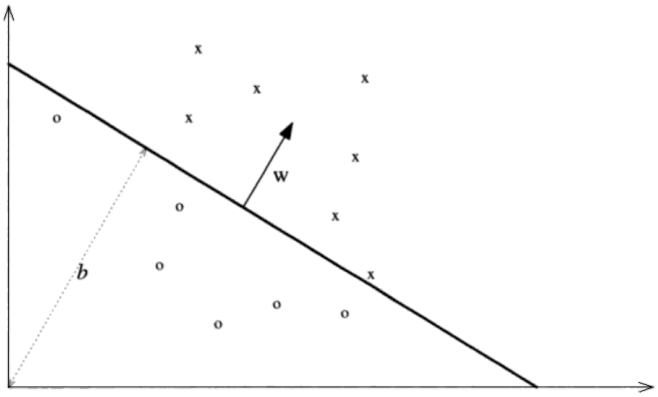


Figure 2.13.: Linear separating hyperplane [62]. The two-dimensional example illustrates that that the linearly separating hyperplane can be described with its normal vector w and a bias b.

For all given instances, the Euclidean distance to this linear function can be calculated, referred to as the margin. This margin is restricted to be positive. Furthermore, the problem can be refined by normalising the weight vector w. From all possible separating hyperplanes, the maximal margin hyperplane needs to found. It is defined to contain the largest margin to all training instances with a minimised norm $\| w \|$ (Figure 2.14).

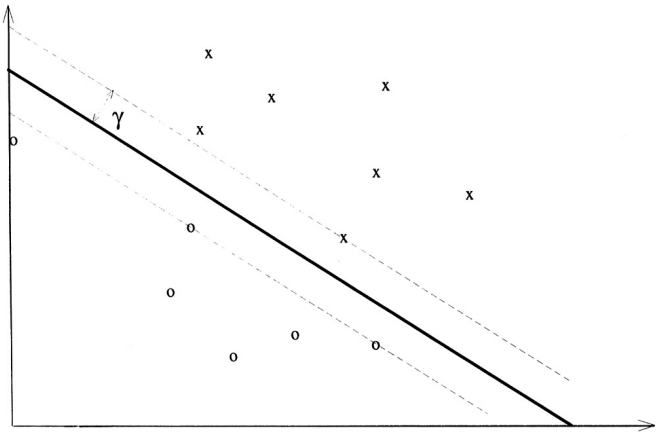


Figure 2.14.: Maximal margin hyperplane [62]. The margin of the points which are lying closest to the hyperplane (smallest distance) is maximised.

The length of a normalised normal vector is equal to one. The points which are closest

to the maximal margin hyperplane are called support vectors and have the functional margin of one, $+1$ for all positive examples and -1 for all negative examples. The maximal margin hyperplane is located exactly in the middle of them.

$$
\begin{aligned}
+1 &= \{w \cdot x^{+}\} + b \\
-1 &= \{w \cdot x^{-}\} + b \\
\gamma &= \frac{1}{2}(\{\frac{w}{\parallel w \parallel_2} \cdot x^{+}\} - \{\frac{w}{\parallel w \parallel_2} \cdot x^{-}\}) \\
\gamma &= \frac{1}{2 \parallel w \parallel_2}(\{w \cdot x^{+}\} - \{w \cdot x^{-}\}) \\
\gamma &= \frac{1}{\parallel w \parallel_2}
\end{aligned}
$$

Considering again a training set with p n-dimensional instances and $Y = \{-1, 1\}$, the weight vector can be expressed as a linear combination of the given examples. The task is to find non-zero values α for the training instances that were introduced as the support vectors. Subsequently, new instances can be classified by computing their sum of inner products with all support vectors and the bias. The classification of positive examples $f(x) \leq 0$ can be expressed with the sign or signum function.

$$
\begin{aligned}
w &= \sum_{i=1}^{p} y_i \alpha_i x_i \\
f(x) &= \langle w \cdot x \rangle + b, \;\; w \,\epsilon\, \mathbb{R}^n, \;\; x \,\epsilon\, \mathbb{R}^n \\
h(x) &= sgn(f(x)) \\
h(x) &= sgn(\langle w \cdot x \rangle + b) \\
h(x) &= sgn(\langle \sum_{i=1}^{p} \alpha_i y_i x_i \cdot x \rangle + b) \\
h(x) &= sgn(\sum_{i=1}^{p} \alpha_i y_i \langle x_i \cdot x \rangle + b)
\end{aligned}
$$

Regarding this definition, finding the support vectors with non-zero α values results in a standard quadratic programming problem which can be solved with the sequential minimal optimisation algorithm (SMO) [62]. The calculation of the inner product with a new input vector can be imagined as computing its similarity to the support vectors. In case of using a linear kernel for classification, this is implicitly defined by

the inner product.

$$a \cdot b = \| a \| \| b \| cos\theta$$
$$cos\theta = \frac{a \cdot b}{\| a \| \| b \|}$$

Geometrically, the inner product of two normalised vectors represents their cosine similarity. Orthogonal vectors result in a cosine equal to zero, because of an angle of 90°. The cosine of angles smaller than 90° is a value between zero and one, where one corresponds to collinear vectors. Angles larger than 90° are generated by a negative cosine. A value of -1 corresponds to opposite vectors.

2.2.4.2. Gaussian Kernel

While the linear kernel was just the inner product of two vectors, the Gaussian kernel measures the similarity between two vectors by generating a score directly from their distance. This score follows a normal distribution.

$$\phi : X \to F$$
$$K(x_i, x) = \langle \phi(x_i), \phi(x) \rangle$$
$$K(x_i, x) = \exp(-\frac{\| x - xi \|^2}{2\sigma^2}) = \exp(-\frac{\sum_{i=j}^{n} (x_j - x_j^{(i)})^2}{2\sigma^2})$$
$$f(x) = \sum_{i=1}^{p} \alpha_i y_i K(x_i \cdot x) + b$$

The training instances can be imagined as landmarks. A new data point which is

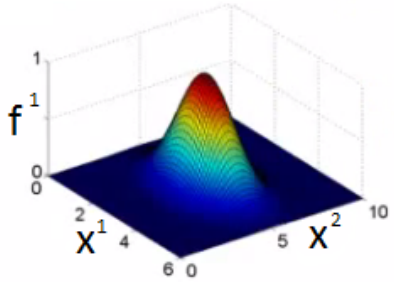


Figure 2.15.: Gaussian kernel. Some two-dimensional training instance is defined with the position $\binom{3}{5}$. The closer a new data point is located to this landmark, the higher its Gaussian score will be.

positioned close to a training instance will result in a low numerator in the exponential function. Therefore, the similarity function will be close to one (Figure 2.15[23,24]). The parameter σ influences the width of the kernel. A small value results in a more narrow shape. The first step is to compute all Gaussian similarities to the training data points for a new example. Subsequently, the decision of the classification ($f(x) \geq 0$) is made by calculating the inner product of the weight vector w and the new feature vector f with a bias b.

$$f(x) = \langle w \cdot f \rangle + b, \quad w \, \epsilon \, \mathbb{R}^p, \quad f \, \epsilon \, \mathbb{R}^p$$

The weight vector of the maximal margin hyperplane using the linear kernel is n-dimensional, because it represents a linear combination of the support vectors ($f(x) = \langle w \cdot x \rangle + b$). In case of the Gaussian kernel, this weight vector is p-dimensional, if all training instances are considered as landmarks. The SVM classification with this kernel is illustrated in Figure 2.16.

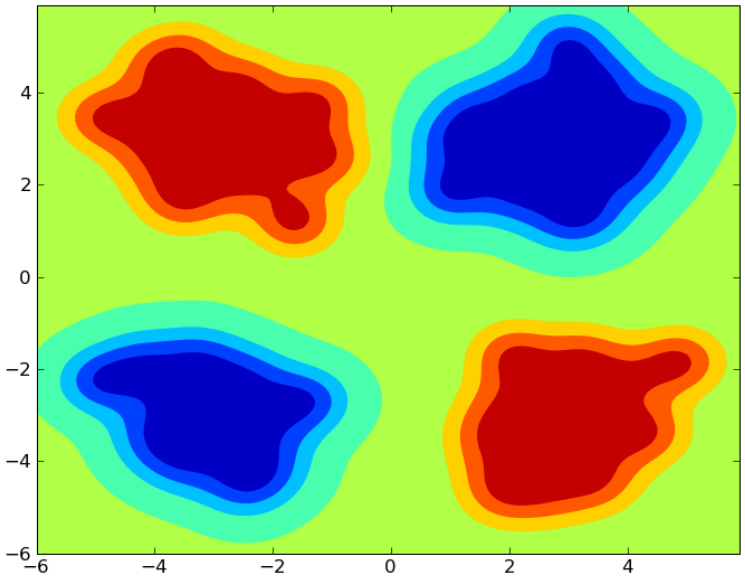


Figure 2.16.: Gaussian kernel XOR classification [20]. In comparison to the classification boarder in Figure 1.2, the Gaussian kernel leads to the round shape of the maximal margin hyperplane surrounding the scatter plots of the two classes.

[23] https://www.coursera.org/instructor/andrewng
[24] https://www.coursera.org/learn/machine-learning

2.2.4.3. Shallow-Linguistic Kernel

Giuliano *et al.* published this kernel to perform relation extraction from biomedical literature [63]. It is defined as the sum of a global context kernel and a local context kernel. These customised kernels were implemented to be executable with the LIBSVM package[25] [64]. Both kernels are explicitly described to work with the inner product of normalised feature vectors.

$$K(x_1, x_2) = \frac{\langle \phi(x_1), \phi(x_2) \rangle}{\| x_1 \| \| x_2 \|}$$

The global context kernel works on unsorted patterns of words up to a length of $n = 3$. These n-grams are implemented using the bag-of-words approach. This method counts the number of occurrences of every word in a sentence including punctuation, but excluding the candidate entities. The patterns are computed regarding the phrase structures fore-between (FB), between (B), and between-after (BA) the considered entities. An example for a protein-protein interaction fore-between pattern is "binding of [P_1] to [P_2]" [63]. Therefore, three row vectors ϕ_P have to computed based on a given training set, one for each of the structures.

$$\phi_P(R) = (tf(t_1, P), tf(t_2, P), \ldots, tf(t_l, P)) \, \epsilon \, \mathbb{R}^l$$

The function $tf(t_i, P)$ counts the number of occurrences of a token t_i in a pattern P. The three l-dimensional vectors will be extremely sparse, because only a few of all available tokens l will be represented in a new classification example. The global context kernel with its three phrase structures was inspired by Bunescu and Mooney [56]. It is implemented by summarising the kernels of the three row vectors.

$$K_{GC}(R_1, R_2) = K_{FB}(R_1, R_2) + K_B(R_1, R_2) + K_{BA}(R_1, R_2)$$

The local context kernel considers tokens with their part-of-speech tags, capitalisation, punctuation, and numerals [9, 63]. The left and right ordered word neighbourhoods

[25] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

up to window size $w = 3$ are considered in two separated kernels ψ.

$$L = t_{-w}, \ldots, t_{-1}, t_0, t_{+1}, \ldots, t_{+w}$$
$$\psi_L(R) = (f_1(L), f_2(L), \ldots, f_m(L)) \, \epsilon \, \{0,1\}^m$$
$$K_{LC}(R_1, R_2) = K_{left}(R_1, R_2) + K_{right}(R_1, R_2)$$

The two vectors contain a value $L = 1$, if a feature is active in the specified position, where m denotes the number of different extracted patterns from the training set. The left hand and right hand side similarity of a new classification example are summed up and added to the result of the global context kernel.

$$K_{SL}(R_1, R_2) = K_{GC}(R_1, R_2) + K_{LC}(R_1, R_2)$$

Therefore, the shallow linguistic kernel measures the similarity of a classification example by computing its feature vectors, calculating all similarities as defined with inner products, and summing them up.

2.2.4.4. All-Path-Graph Kernel

This kernel works on dependency graph representations of sentences, which are gained from constituent or syntax trees [9]. A simple syntax tree for the sentence "SsgG transcription also requires the DNA binding protein GerE." is given in Figure 2.17.

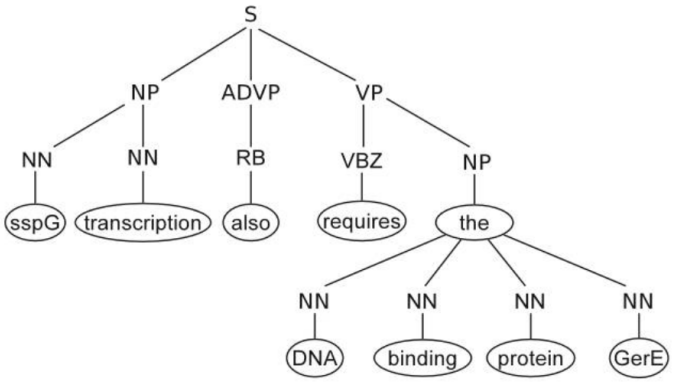


Figure 2.17.: Syntax tree representation [9]. The nodes in the graph represent phrases, namely NP (noun phrase), ADVP (adverb phrase), and VP (verb phrase). NN respresents a singular noun, RB and adverb, and VBZ a verb in the third person singular.

In general, the nodes in the dependency graph representation, also called vertices, are

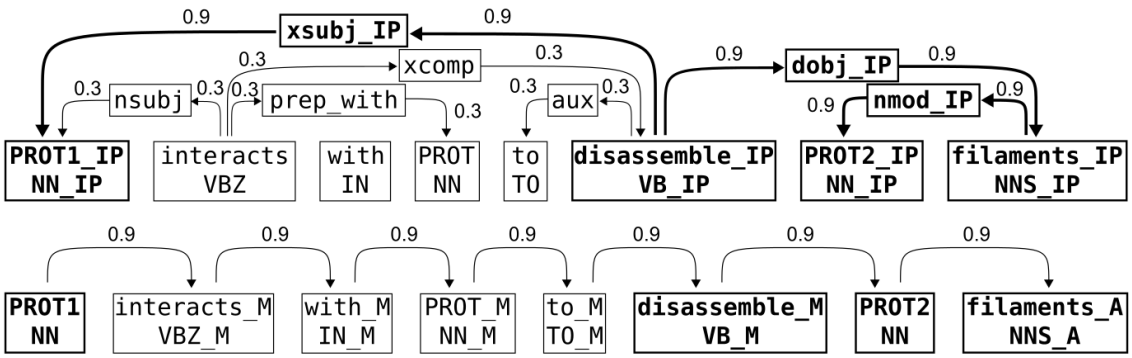


Figure 2.18.: Dependency graph representation [54]. The first subgraph shows the dependencies between the tokens. PROT1 is the subject and the tokens "PROT2 filaments" represent the (direct) object. The shortest path between them is shown in bold and connected by the verb disassemble. Therefore, these tokens are marked with the special tag IP. The phrase between PROT1 and the VP_IP token is described with the label xcomp, representing an open clausal complement. Therefore, the label xsubj represents the controlling subject of the whole sentence, while nsujb names the nominal subject of the complementary phrase. The weights within the shortest path receive the value 0.9 and all other values are set to 0.3. The second subgraph represents the linear order of the sentence. The labels of the tokens also show part-of-speech tags and the text itself, but they additionally contain a positional tag (B)efore, (M)iddle, or (A)fter. These tags are always related to the positions of the considered binary relation of the highlighted entities PROT1 and PROT2. All weights in this subgraph are set to 0.9.

the tokens in the text and the labels show the term itself with a part-of-speech tag. The edges connect the nodes with a weight. The highest weights are given to edges which are part of the shortest path connecting the interaction partners. In this case, the nodes receive a special tag. Furthermore, the text of the tokens representing the candidate entities is replaced with a generic term, e.g. PROT1 and PROT2. A third term PROT can be used for all other potentially related entities. This is also called entity blinding [9]. Each binary example in a sentence is represented by two graphs (Figure 2.18) determining the dependencies[26] and the structure of the phrases. If the pair of selected entities changes, only the labels and the weights will change, not the structure of the graph. All words in the graph are possibly important for the correct prediction of the related entities. Therefore, it can be problematic to omit the words which are not part of the shortest path (Figure 2.19) [54, 56].

Mathematically, all relations in a graph can be represented in an adjacency matrix A.

[26] `http://www.mathcs.emory.edu/~choi/doc/clear-dependency-2012.pdf`: Guidelines for the CLEAR Style Constituent to Dependency Conversion

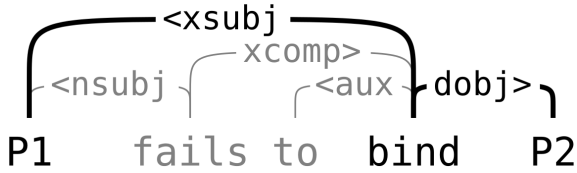


Figure 2.19.: Shortest path [54]. The information that P1 and P2 are actually not interacting is not part of the shortest path. Without considering the complementary phrase shown in grey, this phrase can be considered as "collapsed".

The entries in this matrix determine the weights of the connecting edges. The interesting thing with this representation is that the multiplication of the matrix with itself leads to a matrix with all summed weights of length two[27] [54]. Therefore, all possible paths of all lengths can be calculated by computing the powers of the matrix. Matrix addition of all these matrices results in a final adjacency matrix W, which consists of the summed weights of all possible paths [54]. Paths of length zero are removed by subtracting the identity matrix. This is also called the Neumann Series [54].

$$A \, \epsilon \, \mathbb{R}^{|V| \times |V|}$$

$$(I - A)^{-1} = I + A + A^2 + \ldots = \sum_{k=0}^{\infty} A^k$$

$$W = (I - A)^{-1} - I$$

All labels $\mathcal{L}$ as shown in Figure 2.18 are represented as a feature vector. This feature vector is encoded for every vertex, containing the value 1 for labels that are presented within this particular node. This results in a label allocation matrix L.

$$L \, \epsilon \, \mathbb{R}^{|\mathcal{L}| \times |V|}$$

The feature matrix G is defined by Gärtner *et al.* [65], summing up all weighted paths with all presented labels. This calculation combines the strength of the connection between two nodes with the encoding of their labels (formula (1)). Another possibility is to take the maximum over all weighted paths connecting to labels (formula (2)). Both approaches were implemented by Airola *et al.*[28] [54]. In general, it can be stated that the dependency weights are the higher the shorter their distance to the

[27] `https://1stprinciples.wordpress.com/2008/03/30`
[28] `http://mars.cs.utu.fi/PPICorpora/errata.pdf`

shortest path between the candidate entities is [9].

$$G_{i,j} = \sum_{k=1}^{n} \sum_{l=1}^{n} L_{i,k} W_{k,l} L_{j,l} \tag{1}$$

$$G_{i,j} = \max_{1 \leq k,l \leq n} \{L_{i,k} W_{k,l} L_{j,l}\} \tag{2}$$

The kernel computes the similarity of two matrix representations G' and G'' by summing up the products of all their entries [54].

$$k(G', G'') = \sum_{i=1}^{|\mathcal{L}|} \sum_{i=1}^{|\mathcal{L}|} G'_{i,j} G''_{i,j'}$$

2.2.4.5. Sparse Regularised Least Squares Classifier

The regularised least squares classifier algorithm is similar to a standard support vector machine, but the basic mathematical problem does not need to be solved with quadratic programming. Considering the learning problem formulation, the weight vector is not explicitly defined as a linear combination of training vectors [54]. Instead, a set of basis vectors B is selected in advance from the training data set with m instances.

$$\sum_{j=1}^{m} (y_j - \sum\nolimits_{i \epsilon B} a_i k(x_j, x_i))^2 + \lambda \sum\nolimits_{i,j \epsilon B} a_j a_i k(x_j, x_i)$$

This results in a single system of linear equations which can be solved with the Conjugate Gradient algorithm[29] [66]. The formula contains two terms. The first term represents the squared error or squared loss function [54]. This can be imagined as the function of the output neuron in Section 2.2.3. The second term is important for regularisation. Artificial neural networks and support vector machines can also make use of such a term, but this depends on their implementation. In the case of regularised least squares classification, it is part of the algorithm. The aim is to minimise the given function. If the value λ grows, the weights a_i have to shrink to minimise the output value of this function, iterating over all training instances m. Therefore, a larger value λ provides better generalisation of the classifier. This reduces the risk of memorising the training data set, also called overfitting. The

[29] `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`: An Introduction to the Conjugate Gradient Method Without the Agonizing Pain

classifier was only used for the sparse matrices G of the all-paths graph kernel, but it can be applied to other kernels, too. The implementation used in this thesis [9, 54] selects the decision threshold by computing the weight vector with a given λ, sorting the output scores, and selecting the one with the highest F_1 score. Tikk *et al.* applied the λ values 0.25, 0.5, 1.0, and 2.0, which are also compared in this thesis (Section 3.2).

2.2.5. Entropy-based Methods

Information can be measured with the entropy formula and interpreted as a a value describing the purity of a group of instances [25]. If almost all instances in a data subset belong to one class, the entropy will be close to zero. If both classes are equally represented, the entropy will maximise. This can be illustrated with a decision tree based on the WEKA weather data set [25].

2.2.5.1. Decision Trees

Table 2.5.: Weather data set [25]. The features in this table are outlook, temperature, humidity, and windy. The decision class is play with the binary decision yes or no.

outlook	**temperature**	**humidity**	**windy**	**play**
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 2.5 contains four different features and the binary class to play or not to play. The aim is to build a decision tree that splits the sets of nominal attributes for each

feature in a way that there is only one class represented in each leaf (pure level). Leaves are the last nodes in the tree. All decisions where to split the tree can be made by calculating the entropies for each new node.

$$H = \sum_{i=1}^{I} -\frac{|S_i|}{|S|} \cdot log_2 \frac{|S_i|}{|S|}$$

For each subtree, the logarithm has to be computed for the total number of instances S, divided by the number of instances S_i that encode each class. In the case of a binary classification problem, the information can be measured in bits. The minimum entropy is equal to zero and the maximum value is one, as illustrated in Figure 2.20. Therefore, the logarithm with base 2 is calculated. Considering Table 2.5, the first subtree is the overall tree without any branches. It contains nine times the decision yes and five times the decision no.

$$\begin{aligned} H &= \sum_{i=1}^{I} -\frac{|S_i|}{|S|} \cdot log_2 \frac{|S_i|}{|S|} \\ H &= -1 \cdot ((\frac{9}{14}) \cdot log_2(\frac{9}{14}) + (\frac{5}{14} \cdot log_2(\frac{5}{14})) \\ H &= 0,940 \text{ bits} \end{aligned}$$

The first branching step is calculated by computing the information gain G. The root of the tree will be selected by calculating the entropy of each feature and subtracting it from the overall tree without the new branch. Therefore, the subtree with the smallest entropy will achieve the largest information gain. Considering the branches in Figure 2.20, the feature outlook shows the smallest entropy with 0.693 bits. This number is based on averaging the three entropies of the outlook subtrees. The result is an information gain of 0.247 bits.

$$\begin{aligned} H_1 &= -1 \cdot ((\frac{2}{5}) \cdot log_2(\frac{2}{5}) + (\frac{3}{5} \cdot log_2(\frac{3}{5})) = 0,971 \text{ bits} \\ H_2 &= 0.0 \text{ bits} \\ H_3 &= -1 \cdot ((\frac{3}{5}) \cdot log_2(\frac{3}{5}) + (\frac{2}{5} \cdot log_2(\frac{2}{5})) = 0,971 \text{ bits} \\ H_{outlook} &= \frac{5}{14} \cdot 0,971 \text{ bits} + \frac{4}{14} \cdot 0.0 \text{ bits} + \frac{5}{14} \cdot 0,971 \text{ bits} \\ H_{outlook} &= 0.694 \text{ bits} \\ G_{outlook} &= H - H_{outlook} = 0,940 - 0,694 = 0,246 \text{ bits} \end{aligned}$$

These steps of averaging the entropies and calculating the information gain are repeated for all attribute nodes of the feature outlook, except the leaf overcast, which is already pure. The total entropies for the subsequent calculations and information gain comparisons are 0.971 bits for both attributes, sunny and rainy. The reason is that the sunny node contains two times yes and the rainy node three times yes, but both show a total number of 5 instances. The final decision tree is shown in

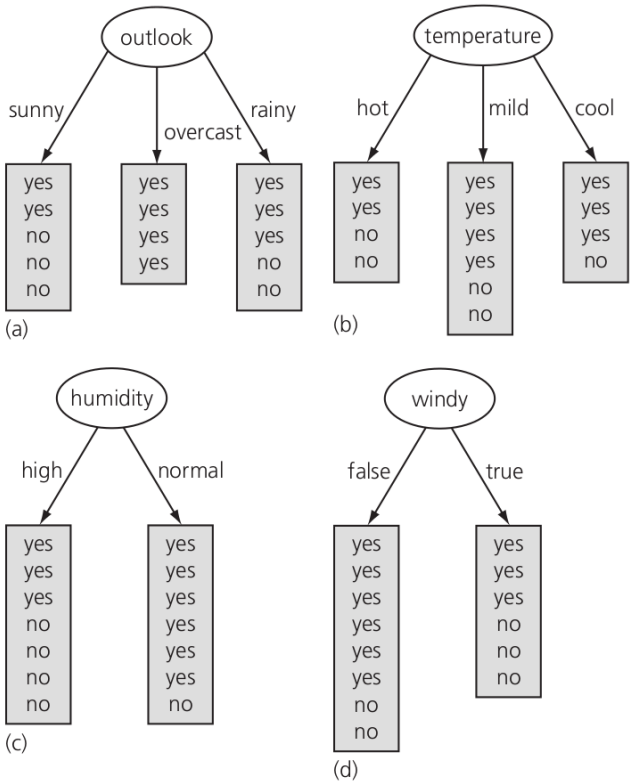


Figure 2.20.: Branching features [25]. All features have to be considered with their averaged entropy. The feature outlook (a) is selected as the root, because its averaged entropy is 0.694 bits. Subtree (b) contains an entropy of 0.911 bits, (c) 0.788 bits, and (d) 0.892 bits. The entropy in the overcast leaf is 0.0 bits and the entropy in the hot leaf is 1.0 bits.

Figure 2.21. One important fact is that all leaves are pure, which is shown in the WEKA output text file (Figure 2.22). Nevertheless, the height of the tree should be considered critically. The aim of reaching pure leaves leads to a risk of overfitting. This process can be controlled with a technique called pruning [25]. The example presented here is calculated with the Iterative Dichotomiser 3 (ID3) algorithm, developed by J. R. Quinlan [25]. While the example shown here is nominal, most classification problems are numerical problems. Therefore, J. R. Quinlan expanded

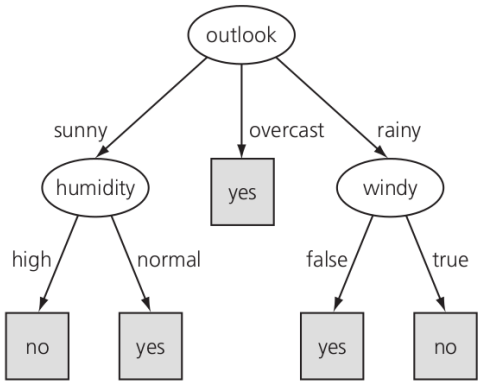


Figure 2.21.: Decision tree [25]. The final decision tree contains the attributes outlook, humidity, and windy. The feature temperature was not selected as relevant from Table 2.5. This is a toy example and the decision to play or not to play cannot be interpreted, because the type of game is not known. The outcome is yes if it is sunny with normal humidity, rainy and not windy, or if the outlook is overcast.

```
outlook = sunny
|   humidity = high: no (3.0)
|   humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)
```

Figure 2.22.: WEKA decision tree [25]. The WEKA tree in text format can be processed with programming approaches and shows the amount of instances in the leaves. The wrongly classified instances are displayed with a slash (not shown here).

his algorithm to deal with numeric attributes, missing values, noisy data, and pruning [25, 67]. It is called C4.5 algorithm[30] and uses binary splits. If there are only binary branches in the tree, it is called a binary decision tree. WEKA uses its own implementation of the C4.5 algorithm, named the J4.8 decision tree. There is also a commercial implementation, called C5.0 [25].

2.2.5.2. Random Forests

A group of random decision trees is called a random forest. These trees perform the branching step at each node, based on a random feature subset of defined size.

[30] `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html`

Furthermore, every random tree is built with a training data set constructed by bootstrap aggregating (drawing with replacement), also called bagging. The reason for this approach is that slight variations in the training data set can change the selected nodes in the tree. Bagging is the attempt to neutralise this instability [25]. All trees are fully constructed with the feature subset (no pruning). This algorithm was invented by L. Breiman [25, 68]. It is implemented in WEKA with the parameters number of features to select, maximum depth of the tree, and number of trees[31]. The decision is made by averaging the output of all trees being zero or one, which represents a majority vote.

2.2.6. Normalisation

There are two different kinds of normalisations, namely scaling features to the same range (formula (1)) and scaling them to unit variance (formula (2)) [69].

$$\tilde{x_1} = \frac{x - l}{u - l} \tag{1}$$

$$\tilde{x_2} = \frac{x - \mu}{\sigma} \tag{2}$$

The difference can be illustrated with an example data set $x = \{200, 300, 400\}$. Considering formula (1), u refers to the upper bound of 400 and l to the lower bound of 200. This results in the values $\tilde{x_1} = \{0, 0.5, 1\}$. Using the second formula with the standard deviation $\sigma = 100$, the result is $\tilde{x_2} = \{-1, 0, +1\}$. Therefore, the first approach scales the values to the range $[0, 1]$. The second approach results in the range $[-1, +1]$, which represents a distribution with zero mean and unit variance.

2.2.7. Dimensionality Reduction

In classification problems, the question arises which features can be considered as most important for the prediction. There are several approaches to select a subset of features from the original data set. WEKA uses two fundamentally different approaches, called filter and wrapper methods. The filter method selects a subset of features, before the training period of the classifier is started. Examples for filter methods are the BestFirst filter and the principal component analysis (PCA). An

[31] `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomForest.html`

example for a wrapper method is the forward selection. The application of these methods reduces the data set complexity and influences the computation time of the classifier's training process positively.

2.2.7.1. Forward Selection

This approach starts with applying the classifier once for each single attribute. The feature with the best prediction result is selected and the whole procedure starts again, combining each attribute once with the feature selected first. The algorithm terminates if the performance stops to increase [25]. This procedure is also called a greedy search. It is able to find a global minimum in the optimisation process, but it can also stop in a local minimum. The risk of local minima can be reduced by using cross-validation, starting multiple runs of a forward selection, or combining it with a backward elimination. This approach of starting with the complete feature set and reducing it by one feature at a time can be considered as the opposite of the forward selection. The combination of forward selection and backward elimination is called a bidirectional search [25]. Another approach is to not only set the increase in the performance as a criterion, but also its amount by a certain percentage [25]. The disadvantage of these wrapper methods is that they are computationally expensive. This effect can increase if the classifier tends to have a longer training period than others, e.g. comparing artificial neural networks and support vector machines.

2.2.7.2. BestFirst Filter

Instead of using the classification algorithm within the feature selection process, the BestFirst filter combines Correlation-based Feature Selection (CFS) with a greedy search, similar to the forward selection presented in the last subsection. WEKA offers the direction options forward, backward, and bidirectional [25]. It also supports a backtracking facility to further evaluate other subsets after a (local) minimum was reached. The algorithm computes the worth or merit of each feature subset by calculating a matrix with all feature-feature and feature-class correlations. The aim is to find subsets of attributes that have a low feature-feature and a high feature-class correlation. As most classification tasks are nominal, e.g. with a positive or negative outcome, the feature-class correlation cannot be measured with the Pearson correlation coefficient. M. Hall shows that discretisation of all continuous features

can be combined with different measurements, similar to the Pearson correlation coefficient [70]. One of the presented ways is to compute the information gain based on entropy calculations. Therefore, the worth of a feature subset S with k features is defined as the heuristic "merit" of the average feature-class correlation $\overline{r_{cf}}$ divided by the average feature-feature correlation $\overline{r_{ff}}$.

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

2.2.7.3. Principal Component Analysis

One of the most popular approaches to reduce the dimension of a feature set is principal component analysis. The idea is to calculate a subset of feature vectors on which the scatter plots can be projected, such that most of the variance in the data set is still covered. This can be illustrated with two-dimensional example (Figure 2.23). The question is how much of the variance can be covered by how many eigenvectors.

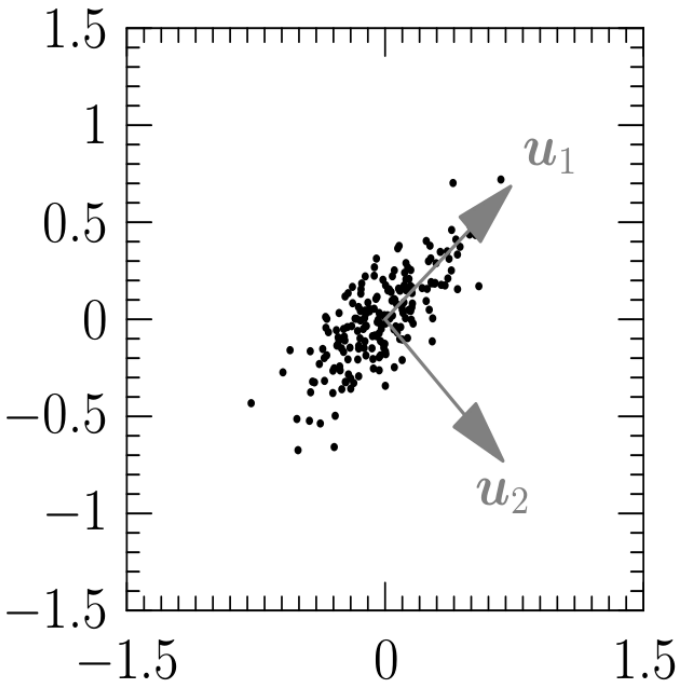


Figure 2.23.: Eigenvectors [71]. The vectors $\vec{u_1}$ and $\vec{u_2}$ represent the eigenvectors of this scatterplot. The first vector shows the largest variance.

This can be calculated with the covariance matrix Σ. It contains all covariances

between the features. Therefore, the variances are located on the diagonal.

$$Var(x) = \frac{\sum_{i=j}^{n}(x_i - \bar{x})(x_i - \bar{x})}{n-1}$$
$$Cov(x,y) = \frac{\sum_{i=j}^{n}(x - \bar{x})(y - \bar{x})}{n-1} = \Sigma$$

An eigenvector u is defined as a vector that can be multiplied with a matrix (here Σ), resulting in a vector with the same direction. The length of this output vector represents the variance of the data set in this direction. The factor or scalar λ by which the length of this vector differs from the eigenvector is called eigenvalue. Therefore, it is important to perform feature scaling before computing the principal components, if the feature ranges are divers.

$$\Sigma u = \lambda u$$

The eigenvectors and eigenvalues can be found by diagonalising the covariance matrix, a method called eigenvalue decomposition [71]. Considering the $n \times n$ covariance matrix Σ, the resulting matrix U contains n eigenvectors as columns and the matrix S contains all eigenvalues sorted by size on its diagonal. The overall amount of variance, that is covered by taking a subset of k eigenvectors to form the $n \times k$ matrix U_{reduce}, can be calculated by computing the projection vectors z and the squared distances of the orthogonally projected coordinates of x_{approx} to the input vectors x over all instances p. If the output value is smaller than 0.01, it means that above 99 % of the variance is covered[32].

$$z = U_{reduce}x$$
$$x_{approx} = U_{reduce}z$$
$$0.01 \geq \frac{\frac{1}{p}\sum_{i=j}^{p} \parallel x^{(i)} - x_{approx}^{(i)} \parallel^2}{\frac{1}{p}\sum_{i=j}^{p} \parallel x^{(i)} \parallel^2}$$

One of the commonly used explanation examples is Fisher's Iris data set, e.g. implemented with visualised principal components in the Python machine learning

[32] `https://www.coursera.org/learn/machine-learning`

library scikit[33]. The Iris flowers were categorised by R. A. Fisher in 1936 with sepal and petal lengths and widths [72]. The data set can be visualised as separated scatter plots by computing the principal components.

2.2.8. The WEKA Library

WEKA is an open source software "workbench" from the University of Waikato, New Zealand (Waikato Environment for Knowledge Analysis)[34]. The evaluation approaches, preprocessing methods, and machine learning algorithms explained in this chapter, including several more, are implemented in WEKA. The Explorer Environment can be used to perform calculation steps presented in this chapter one after the other [25]. The start menu of the WEKA Explorer user interface shows basic information, such as the number of instances and the names of the descriptors, illustrated in Figure 2.24. In this case, the selected descriptors refer to molecular substructures of chemical substances. The data set referred to here is further explained in Section 3.3, where it is shown how single workflow steps can be connected to a pipeline using the KnowledgeFlow Environment. All selected components can also be computed and combined by using the Java open-source library with the command-line interface [25].

[33] http://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html
[34] http://www.cs.waikato.ac.nz/ml/index.html

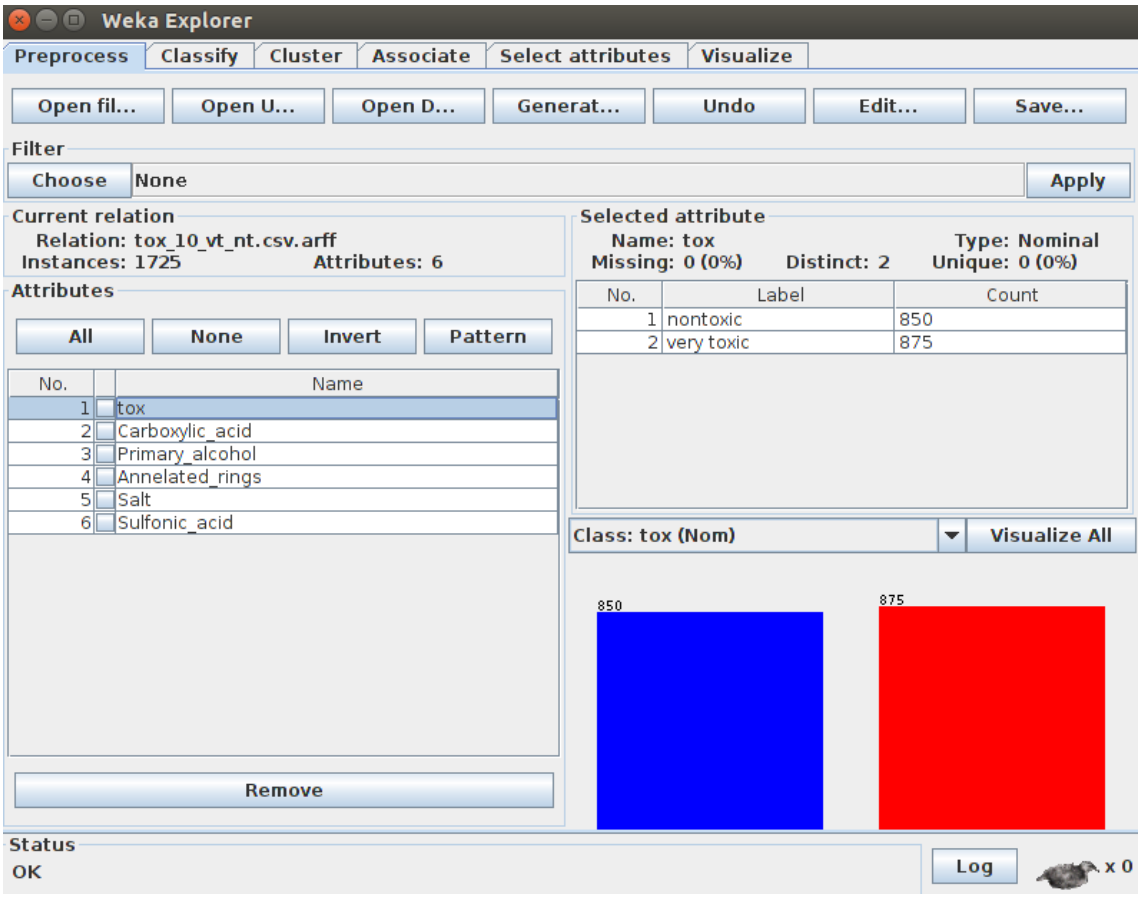


Figure 2.24.: WEKA Explorer Environment. The data set presented here was converted from CSV format to the WEKA specific ARFF format. The amount of instances of each class and the names of the descriptors are shown. Preprocessing can be applied by clicking the *Choose* button, e.g. to apply a principal component analysis. In this case, a BestFirst selection was already performed with the select attributes register card. The computation step of classification can be chosen analogously. All available parameters for each classifier in WEKA can be changed by clicking the submenus.

2.3. Cheminformatics

This section covers a small part of methods from the research area of cheminformatics. The distinction from the area of bioinformatics is fuzzy, but it can be said that "Chem(o)informatics is a generic term that encompasses the design, creation, organization, management, retrieval, analysis, dissemination, visualization, and use of chemical information"[35] and that "Chemoinformatics is the application of informatics methods to solve chemical problems" [73].

2.3.1. Simplified Molecular Input Line Entry Specification

Information about the structure of a molecule can be saved in Structure Data Format (SDF) files[36] or in a SMILES string, the abbreviation for Simplified Molecular Input Line Entry Specification[37]. The SDF file contains atomic coordinates with types of bonds and probably some meta information like in the PubChem SDF files[38]. A SMILES string consists of atoms and bonds with numbers or brackets[39]. Numbers are used to close a ring structure like the one shown in Figure 2.25. Round brackets open and close a branched substructure shown in Figure 2.26. A combination of numbers and brackets in a SMILES structure is shown in Figure 2.27. The figures were generated with MarvinSketch[40]. The conversion of SDF files to the SMILES format and vice versa is illustrated in Section 3.4.

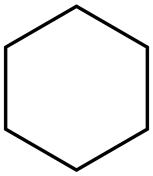


Figure 2.25.: Cyclohexane. This structure was generated with the SMILES C1CCCCC1. The number after the first C atom determines the start of the ring structure and the number after the sixth C atom closes this structure.

[35] G. Paris, August 1999, ACS meeting
[36] http://www.epa.gov/ncct/dsstox/MoreonSDF.html
[37] http://www.daylight.com/meetings/summerschool98/course/dave/smiles-intro.html
[38] https://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch-help.html
[39] http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html
[40] https://www.chemaxon.com/products/marvin/marvinsketch

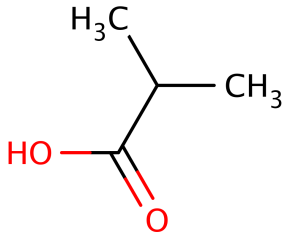


Figure 2.26.: Isobutyric acid. The SMILES CC(C)C(=O)O was used to display this structure. The first round pair of brackets represents the branched methyl group and the second pair contains the double-bonded oxygen.

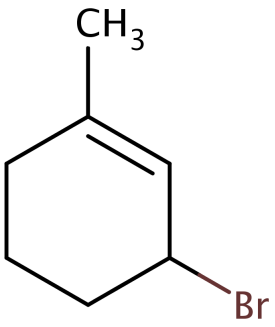


Figure 2.27.: 1-Methyl-3-bromo-cyclohexene-1. Numbers as well as brackets are contained in the SMILES CC1=CC(Br)CCC1. The generated structure shows the order of atoms in this pattern, starting with the methyl group. The ring structure is opened with the first number. The C atom after the double bond of the hexene is followed by a bromine atom in brackets, because it branches from the ring structure. After another three carbon atoms, this ring is closed with the last number. The SMILES CC1=CC(CCC1)Br leads to the same structure. Therefore, the ring closure can also be considered as the explicitly branched structure. This can be useful in case of an ongoing chain, replacing the Br branch.

2.3.2. SMiles ARbitrary Target Specification

This format is based on SMILES and abbreviated with SMARTS. It is used to search databases for substructures. Almost all SMILES strings are also SMARTS strings, because both cover atoms and bonds in their format. The SMARTS format is extended with symbols to generalise the string representation, allowing e.g. two different atoms at a position[41]. The SMARTSviewer server[42] offers a good visualisation of SMARTS and their meaning in a detailed legend as shown in Figure 2.28 and Figure 2.29. SMARTS patterns can be generated with OpenBabel[43]. In combination with physicochemical properties, also supported by OpenBabel, these molecular descriptors can be used as machine learning features, described in Section 3.3. Other features used in this thesis were generated with the Schrödinger's Maestro software QikProp in combination with LigPrep[44].

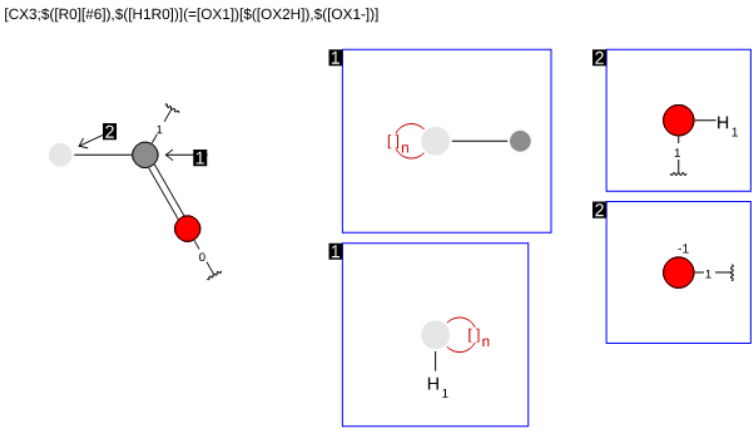


Figure 2.28.: Carboxylic acid SMARTS [51].
The SMARTSviewer displays a detailed view on the SMARTS pattern [CX3;$([R0][#6]),$([H1R0])](=[OX1])[$([OX2H]),$([OX1-])] from two different views in space. This SMARTS string includes carboxylate anions.

[41] http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html
[42] http://smartsview.zbh.uni-hamburg.de
[43] https://github.com/openbabel/openbabel/blob/master/data/SMARTS_InteLigand.txt
[44] http://www.schrodinger.com/Maestro

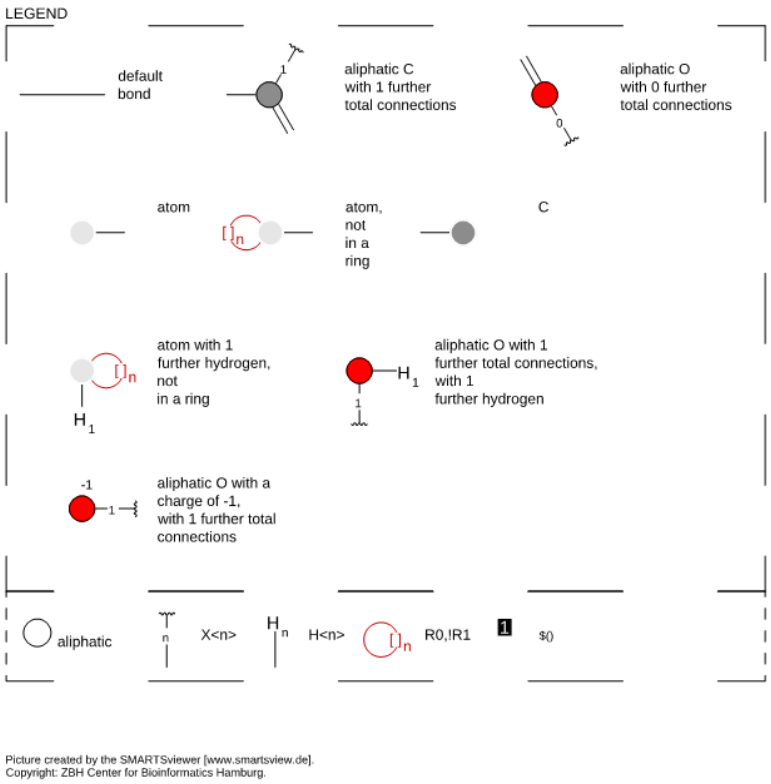


Figure 2.29.: SMARTS legend [51]. The SMARTSviewer legend gives explanations about the different views shown in Figure 2.28.

2.3.3. Toxicity Data Set

ChemIDplus[45,46] is free Web search system which provides information about the toxicity of substances and partially also about their effects. This thesis focusses on LD_{50} data, based on intravenous application in mice. LD is the abbreviation for lethal dose and the value 50 defines the probability with which an organism will get killed by a single dose of a substance [74]. The LD_{50} values are provided in mg per kg body weight (mg/kg bw). ChemIDplus offers a menu for data selection (Figure 2.30). A set of molecules is displayed with links providing detailed information as shown in Figure 2.31 and Figure 2.32. In many cases, there are also references to other types of experiments (Figure 2.33). A substance can be identified with its Chemical Abstracts Service (CAS) number[47]. The corresponding SMILES pattern can be downloaded from ChemIDplus and processed as indicated in Section 2.3.2.

[45] `www.chem.sis.nlm.nih.gov/chemidplus`
[46] `http://www.nlm.nih.gov/pubs/factsheets/chemidplusfs.html`
[47] `https://www.cas.org/content/chemical-substances/faqs`

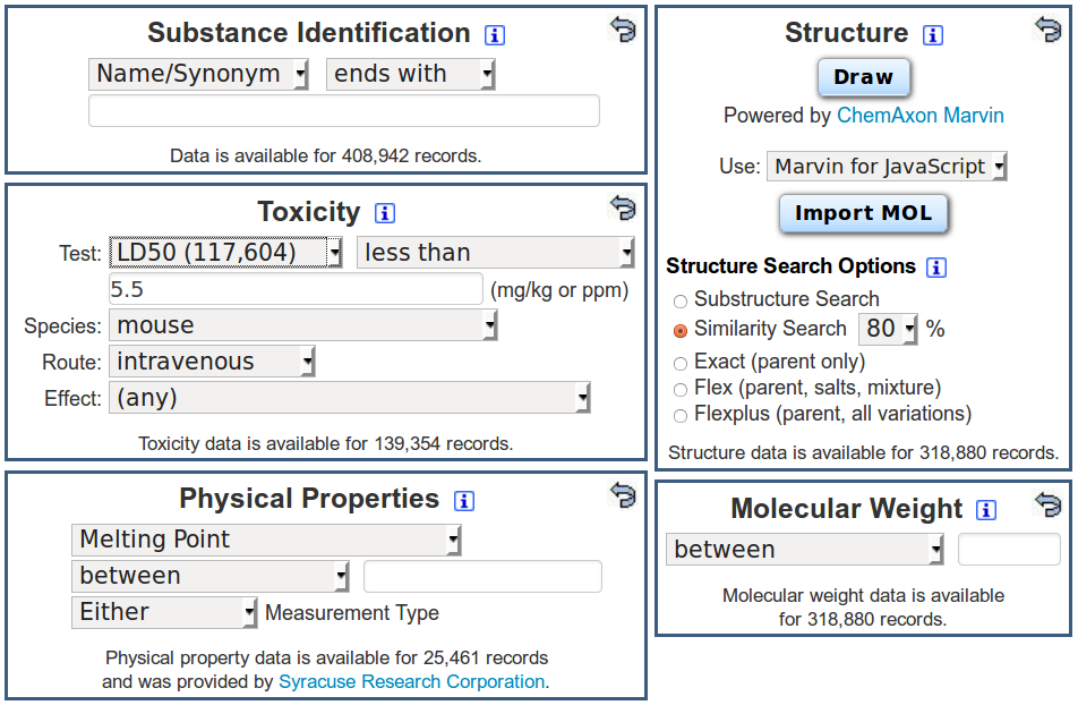


Figure 2.30.: ChemIDplus selection. The selection menu on the ChemIDplus start page can be used to specify organism, applied test, and dose of the substance.

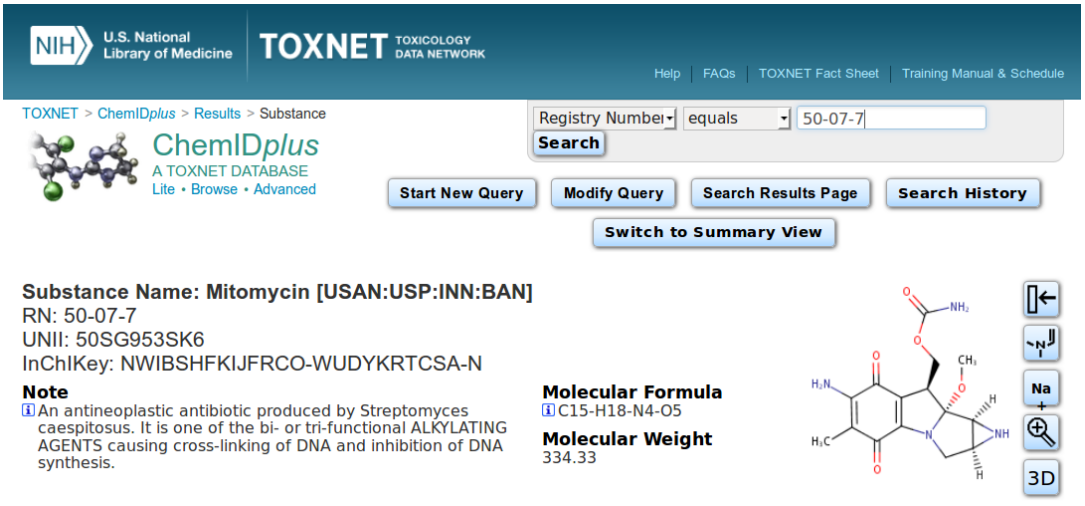


Figure 2.31.: Mytomycin on ChemIDplus. A molecule can be searched directly with the CAS number (here 50-07-7).

Registry Numbers

CAS Registry Number
50-07-7

FDA UNII
50SG953SK6

Other Registry Numbers
144085-53-0 74349-48-7 7481-68-7

System Generated Number
0000050077

Structure Descriptors

InChI
1S/C15H18N4O5/c1-5-9(16)12(21)8-6(4-24-14(17)22)15(23-2)13-7(18-13)3-19(15)10(8)11(5)20/h6-7,13,18H,3-4,16H2,1-2H3,(H2,17,22)/t6-,7+,13+,15-/m1/s1
Download

InChIKey
NWIBSHFKIJFRCO-WUDYKRTCSA-N
Search the web for this InChIKey

Smiles
N12C=3C(C(C)=C(C(C3[C@H]([C@@]2([C@H]2[C@H](C1)N2)OC)COC(N)=O)=O)N)=O
Download

Figure 2.32.: ChemIDplus search result. Mytomycin, as shown in Figure 2.31, can be downloaded as a SMILES structure from ChemIDplus.

mouse	LD50	intraperitoneal	4mg/kg (4mg/kg)		Journal of Antibiotics, Series A. Vol. 13, Pg. 27, 1960.
mouse	LD50	intravenous	4mg/kg (4mg/kg)		Journal of Antibiotics, Series A. Vol. 13, Pg. 27, 1960.
mouse	LD50	oral	23mg/kg (23mg/kg)	BEHAVIORAL: SOMNOLENCE (GENERAL DEPRESSED ACTIVITY) GASTROINTESTINAL: "HYPERMOTILITY, DIARRHEA"	Cancer Research. Vol. 20, Pg. 1354, 1960.
mouse	LD50	subcutaneous	7300ug/kg (7.3mg/kg)		Japanese Journal of Cancer Research. Vol. 80, Pg. 670, 1989.
mouse	LD50	unreported	12mg/kg (12mg/kg)		Cancer Research. Vol. 46, Pg. 2703, 1986.
mouse	LDLo	intratracheal	4mg/kg (4mg/kg)		Toxicology Letters. Vol. 30, Pg. 63, 1986.

Figure 2.33.: Lethal dose of mytomycin. Different routes of administration are shown for this chemical substance, including the source of information. LDL_0 is the lowest dose of a substance reported to have caused death in humans or animals.

3. Results

3.1. PubMed2Go

Parts of this section are similar or identical to a recently submitted publication in BMC Bioinformatics, Softare articles, with me as first author [75].

3.1.1. Related Work

3.1.1.1. Providing Software Interoperability

Several proposals have been made to deal with the problem of interoperability between different software solutions for natural language processing. Only a few of them can be considered as used by the community, namely the Unstructured Information Management Architecture (UIMA) [76, 77], the General Architecture for Text Engineering (GATE) [78], and the BioC XML data format [79].

UIMA supports Text Analysis Engines (TAEs), the text processing software modules, and a common analysis structure (CAS), the XML-based input and output format for TAEs [79]. U-Compare is a Java Web Start application that offers drag-and-drop construction of workflows for UIMA-compatible natural language processing tools [80].

The GATE Developer is an integrated development environment in Java similar to U-Compare [76]. GATE provides an interface to UIMA as well [79].

BioC uses a minimalistic approach, as only the structure of BioC XML files is defined in a document type definition (DTD) file and the user-specific semantics of data and annotations, which are described in a key file. The interoperability is ensured within the BioC workflow, defining an Input Connector to read and an Output Connector

to write BioC XML data. The interface to these BioC classes is implemented in several programming languages [79, 81].

Rak *et al* . claimed that there is a tendency towards workflow construction platforms, but that their software dependencies on a source platform can restrict the development process [76]. Therefore, Web services became popular to solve natural language processing tasks, especially because of the Representational State Transfer (REST) architecture. This was their motivation to develop Argo, an online text mining workbench combining different data formats like CAS and BioC, with the ability to use other Web services in a workflow [76]. There are advantages and disadvantages for choosing UIMA or BioC, but the integration of UIMA-compatible modules can be considered as a more complex process than extending the standard XML format to the BioC XML DTD structure [81]. The aim of PubMed2Go is to enable users to develop text mining applications and to generate use cases with a very basic programming knowledge. This is understood in terms of a stand-alone application without dependency on Web services, but with the possibility to query them if desired. Therefore, the software implements a BioC interface. Any tool in a customised workflow supporting the BioC input and output modules can be used to perform natural language processing tasks independently, as shown in Section 3.1.3.

3.1.1.2. Processing PubMed

At the beginning of 2015, PubMed consisted of more than 24 million records containing 13.1 million abstracts, and the number increases quickly. Considering the issue of how to deal with this large amount of data and to apply natural language processing methods effectively, quite a few published as well as unpublished approaches exist.

There are efforts to simplify literature searches in PubMed and to support text annotation. Two of the most recent Web service developments are OntoGene [13] and PubTator [82], which provide a BioC interface.

Working with PubMed publications on a local machine is possible after downloading the XML files described in Section 2.1.1. The uncompressed size of the complete PubMed XML data set is 114 GB[1]. The user can apply the NCBI interface introduced in Section 2.1.1 to download a set of PubMed XML files related to a specific search term. Biopython[2] can be used to connect to this interface, named EFetch. It also

[1] `http://www.nlm.nih.gov/bsd/licensee/2015_stats/baseline_doc.html`
[2] `http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc120`

contains a library to parse PubMed XML files. Mining XML files can be time-consuming if the parsed results are not saved in a database, such that they have to be processed multiple times. The LingPipe project[3] is implemented in Java and offers a library to parse XML files and build a full text index with Lucene[4]. The documentation contains a short tutorial about loading the abstract and title texts into a MySQL relational database in a version from 2010. It is noteworthy that the PubMed XML schema is updated annually. Biopython and LingPipe are considered as sophisticated frameworks, but this also means that it is a complex task to modify existing and develop new functions to process and index all recently available PubMed XML attributes.

There are also finished implementations building a relational database from XML files. According to the tendency towards Web services as mentioned by Rak *et al* . [76], these approaches were published around 10 years ago [83, 84]. Yoo *et al* . describe a complex system for downloading PubMed and PMC articles in XML format, storing them in a MySQL relational database, and searching the documents with a Lucene full text index [84]. Unfortunately, their service is not hosted anymore. In 2004, Oliver *et al* . compared different approaches for loading PubMed XML files into a relational database [83]. It took 196 hours to run a Java SAX parser on 396 XML files with an uncompressed size of 40.8 GB and to load them into the relational database Oracle 9i. Their second effort was to use PERL code in combination with Oracle with parallelised batches of 50 XML files, which resulted in a runtime of 132 hours. They also used a Java parser in combination with IBM's DB2 database on 500 XML files (number of PubMed IDs unknown), which took 76 hours. The reason they chose to use Oracle or DB2 instead of an open-source PostgreSQL relational database was that they experienced a faster data upload with these technologies and a more effective keyword search engine at that time [83]. There is an unpublished update version from 2010 using Java 6 and MySQL 5.1[5]. In this version, the upload SQL statements for processed XML elements still can be executed separately or included in the Java code. In PubMed2Go, this SQL schema is completely adapted and slightly modified, but combined with object-relational mapping (ORM, Section 2.1.2) in Python to generate a PostgreSQL database from PubMed XML files. That means, changes of SQL tables or columns can be introduced

[3] `http://alias-i.com/lingpipe-3.9.3/demos/tutorial/medline/read-me.html`
[4] `http://lucene.apache.org`
[5] `https://simtk.org/home/medlineparser`

directly in the parser itself, and the whole uploading process can be upscaled to the number of desired CPU cores with multiprocessing. Meanwhile, a text search engine has been developed for PostgreSQL[6]. PubMed2Go implements a Xapian full text index, similar to Yoo *et al.* [84], as any PostgreSQL column can be indexed with this technology using only a few functions. Currently, abstract titles and texts, MeSH terms, keywords, and chemical substances are indexed in the standard implementation. There is also a modified version only indexing abstract titles and texts. Using a Xapian index offers fast and straight forward keyword and context search, as shown by the use cases described in Section 3.1.4. By combining the given approaches, it becomes easy to develop text mining applications based on a local version of PubMed without additional programming efforts.

While PubMed2Go was tested in Ubuntu and Fedora, there is also a one-click-solution based on Docker[7], a system similar to a virtual machine, so that the relational database and the full text index can be used in many more operating systems. Therefore, PubMed2Go helps to standardise the way of processing large literature data sets locally to apply natural language processing models via the BioC interface.

3.1.2. Basic Workflow

The basic requirements are standard installation of Python, Xapian, and PostgreSQL, as well as a minimum of 282 GB free disk space in case of processing all downloadable XML files from PubMed. Figure 3.1 shows the general workflow for loading PubMed XML files into a PostgreSQL relational database and generating a full text index with Xapian.

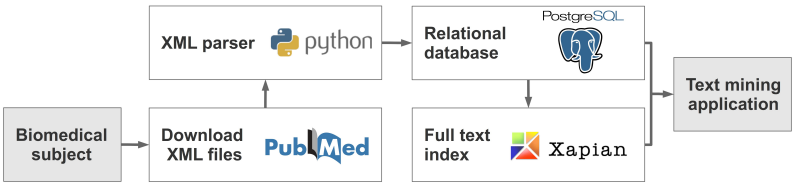


Figure 3.1.: PubMed2Go workflow. 1) Download XML files from PubMed. 2) Parse and upload data into a PostgreSQL relational database. 3) Build a Xapian full text index. 4) Develop text mining applications (use cases in Figure 3.2)

[6] http://www.postgresql.org/docs/9.4/static/textsearch-intro.html
[7] http://docs.docker.com/installation

PubMed2Go is usable via a command-line interface and requires PubMed XML files as input. A PostgreSQL database needs to be created and configured at first. The tables are built based on user-provided data. The SQL schema can be seen in the Supplemental Material. All PubMed XML attributes are transformed into PostgreSQL tables and columns with an object-relational mapping approach using a SAX parser (Section 2.1.2). After processing the PubMed XML files, a Xapian full text index is built by querying titles, abstracts, MeSH terms, keywords, and chemical substances from PostgreSQL. These installation steps can also be executed at once using the virtual container Docker without installing additional software packages. This system is similar to a virtual machine and easy to deploy.

Based on this data environment, a phrase search can be executed with user-defined terms. Furthermore, there is the possibility to send conditional queries to the full text index with AND, OR, NOT, and NEAR. The identified keywords and PubMed IDs are stored in a CSV file. Installation instructions, scripts performing BioC annotations and the generation of graphical plots, or SQL queries as shown in Table 3.1 illustrate how to use the PubMed2Go framework. They are precisely described on the GitHub project page[8].

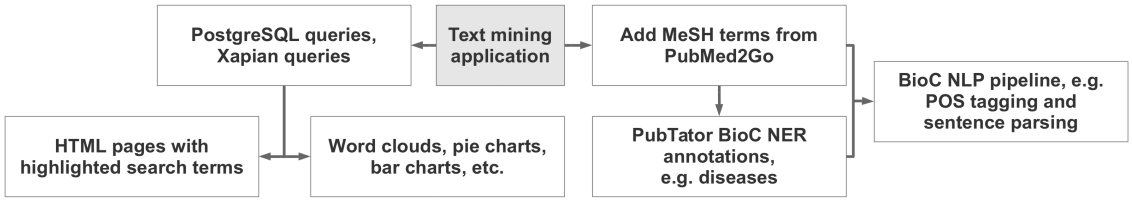


Figure 3.2.: Text mining applications. PubMed2Go can be extended with use cases such as HTML pages with highlighted search terms, word clouds as shown in Figure 3.5 and Figure 3.6, and a bar chart or pie chart as displayed in Figure 3.7 and Figure 3.8. Furthermore, the framework offers the possibility to be extended with natural language processing tools by applying the BioC workflow, as illustrated in Figure 3.3.

3.1.3. Use Case: BioC Applications

The PubMed2Go BioC XML interface can be used to build and interconnect natural language processing models and methods, e.g. the BioC natural language processing pipeline from Comeau *et al* . (right part of Figure 3.2) [85]. As mentioned in the

[8] `https://github.com/KerstenDoering/PubMed2Go`

introduction, the requirements for BioC are a DTD file[9] to define the XML document structure and a key file[10] to explain which structural elements are used in a particular BioC XML document. The infon elements are used to differentiate between distinct XML elements. Every infon element contains a key and a value. This basic element can refer to the document sections title or text, to token IDs, or to named entity recognition annotations.

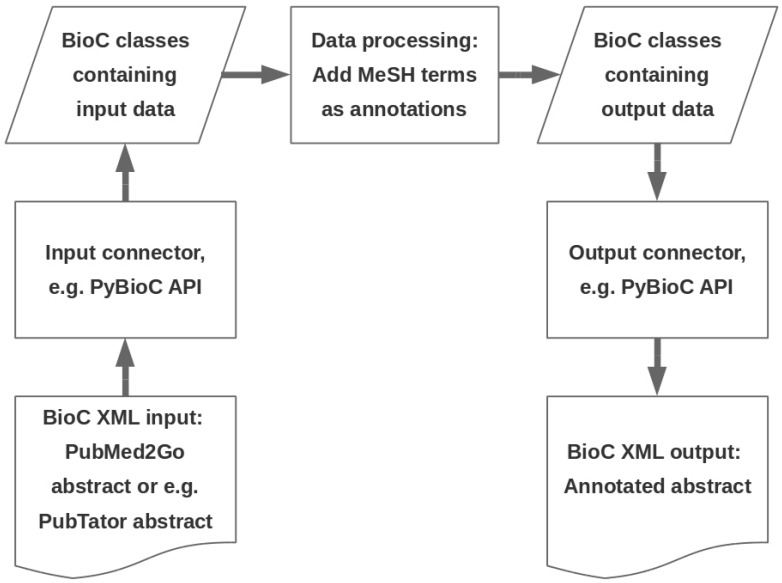


Figure 3.3.: BioC workflow. The minimalistic approach from Comeau *et al.* [79] is modified with the example how to add MeSH terms to BioC PubMed titles and abstracts from the PubMed2Go PostgreSQL database or e.g. to a PubTator-annotated abstract.

By exporting the BioC XML format from the PubMed2Go PostgreSQL database for abstract titles and texts, it becomes possible to use the Web service PubTator [82], the named entity recognition stand-alone tools from the tmBioC package [81], or the BioC natural language processing pipeline [85], as illustrated in Figure 3.2. By applying these tools, named entity recognition of chemicals, diseases, genes, mutations, and species will be included as well as tokenisation, part-of-speech tagging, and sentence parsing. Based on these methods, even more sophisticated natural

[9] `https://github.com/KerstenDoering/PubMed2Go/blob/master/BioC_export/BioC.dtd`
[10] `https://github.com/KerstenDoering/PubMed2Go/blob/master/BioC_export/Explanation.key`

language processing approaches can be applied such as kernel methods used for the identification of PPIs [9]. In case of PubTator and tmBioC, the plain text input format can be combined with a BioC XML output, too. Thereby, the BioC workflow as shown in Figure 3.3 implies that any tool supporting this input and output format can add annotations to a document, supporting the idea of interoperability. The example shows that MeSH terms are added to a PubMed2Go XML document or an article that was annotated with disease terms by PubTator before. Considering the latter case, the XML annotation elements can be merged as shown in Figure 3.4. In this XML schema, a collection consists of PubMed articles with their PubMed ID as document ID. The basic document consists of a title and the abstract text, if given. The original creation source is the PubTator Web service with a document creation date. The file PubTator.key describes the semantics used by this Web service. As named entity recognition tools provided by PubTator are based on the tmBioC package [81], their file tmBioC.key can be used as well. In Figure 3.3, the PyBioC library enables the usage of BioC classes in Python [86], but programming languages such as C++, Java, or Ruby can be used as well[11]. Comeau *et al* . intend that only one type of annotation should be used with one BioC XML file consisting of several documents [79], but they can also be merged as shown here. The only limitation is that the PubTator key file differs slightly from the PubMed2Go key file in terms of semantics, but this can be modified easily by joining the explanations in the key file.

More BioC-related software and text corpora can be found in the overview of the BioCreative IV interoperability track [87]. These approaches illustrate how the BioC interoperability can be used for fast development of prototypic text mining applications in terms of software modularity.

3.1.4. Use Case: Querying PubMed2Go Data Sets

This section describes ideas how to analyse PubMed2Go data sets (left part of Figure 3.2). The PubMed2Go documentation refers to a small data set of 272 MB[12], related to pancreatic cancer, which can be processed in a few minutes. Pancreatic cancer is one of the most dangerous cancer types. Currently, the only way to cure a patient is surgery, beside several therapeutic strategies that cannot significantly increase survival rates [88]. The research progress in this area can be supported with

[11] `http://bioc.sourceforge.net`
[12] `https://github.com/KerstenDoering/PubMed2Go/wiki`, 16 April 2015

```
−<collection>
   <source>PubTator</source>
   <date>2015/04/09 </date>
   <key>PubTator.key</key>
  −<document>
     <id>1000475</id>
    −<passage>
       <infon key="type">title</infon>
       <offset>0</offset>
      −<text>
         Carcinoembryonic antigen (CEA) activity in pancreatic juice of patients with pancreatic carcinoma and pancreatitis.
       </text>
      −<annotation id="0">
         <infon key="MEDIC">D010190</infon>
         <infon key="type">Disease</infon>
         <location offset="77" length="20"/>
         <text>pancreatic carcinoma</text>
       </annotation>
      −<annotation id="1">
         <infon key="MEDIC">D010195</infon>
         <infon key="type">Disease</infon>
         <location offset="102" length="12"/>
         <text>pancreatitis</text>
       </annotation>
      −<annotation id="0_MeSH">
         <infon key="type">MeSH term</infon>
         <location offset="0" length="24"/>
         <text>Carcinoembryonic Antigen</text>
       </annotation>
      −<annotation id="1_MeSH">
         <infon key="type">MeSH term</infon>
         <location offset="43" length="16"/>
         <text>Pancreatic Juice</text>
```

Figure 3.4.: Part of a BioC XML document. The document ID 100475 is its PubMed ID. PubTator annotations are shown with infon elements that contain the key type with the value Disease and the key MEDIC referring to a MeSH ID, like D010190 for the given disease pancreatic carcinoma. The PubMed2Go MeSH term annotations are shown with the annotation IDs 0_MeSH and 1_MeSH to make them distinguishable from the normally iterating PubTator annotation IDs.

text mining methods, e.g. by covering findings about gene-disease and compound-protein relationships.

To get a first impression of important genes or proteins, drugs, and diseases related to pancreatic cancer, selected search terms[13] were manually extracted from DrugBank [89] and OMIM[14]. These databases are focused on human diseases, their interrelated gene mutations, and how to treat them. If e.g. specific oncogenes like KRAS are presented in OMIM, a Xapian search should result in a reasonable number of publications. The relative frequencies of such search terms are displayed in a word cloud in Figure 3.5. The numbers of abstracts identified for each synonym were transformed to a logarithmic scale to smooth differences in word sizes. Pancreatic ductal adenocarcinoma is the most frequently appearing type of pancreatic cancer

[13] `https://github.com/KerstenDoering/PubMed2Go/blob/master/full_text_index/synonyms`

[14] `http://omim.org`

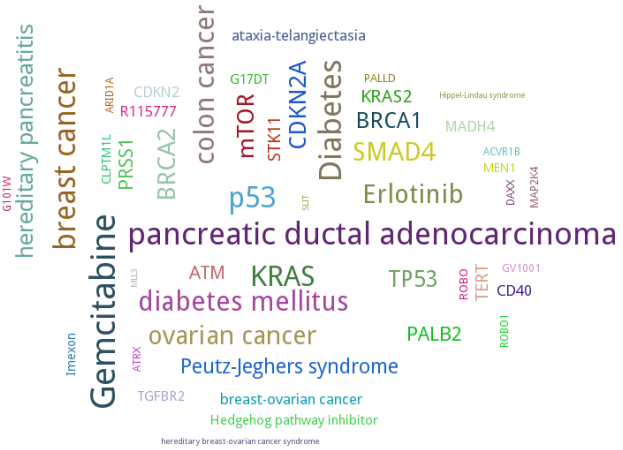


Figure 3.5.: Genes, proteins, drugs, and diseases related to pancreatic cancer. This word cloud shows selected search terms from DrugBank and OMIM with their relative frequencies in a logarithmic scale.

and the most common lethal cancer [90]. Therefore, it is reasonable to see this term displayed as the largest word in Figure 3.5 together with gemcitabine, a nucleoside analogue commonly used in chemotherapy[15]. Indeed, KRAS is the largest displayed gene beside p53. Figure 3.5 also illustrates that texts referring to pancreatic cancer are also related to breast, colon, ovarian, and lung cancer, as well as diabetes, pancreatitis, and other diseases.

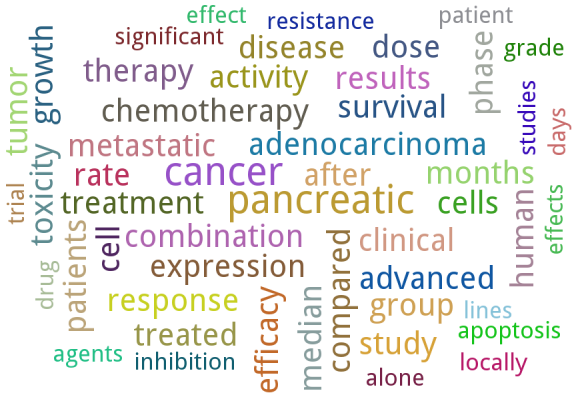


Figure 3.6.: Most frequently co-occurring terms with gemcitabine. These are the 50 most commonly used words surrounding the search term gemcitabine in the pancreatic cancer data set. The identified texts were tokenised. These word frequencies were transformed to a logarithmic scale. No word stemming was performed.

[15] `http://www.drugbank.ca/drugs/DB00441`

The word cloud in Figure 3.6 was generated by processing all abstracts and titles which were found with the drug name gemcitabine to reveal part of the vocabulary related to pancreatic cancer. A whitespace tokeniser was used that separates words as tokens and removes punctuation. After excluding stop words[16] from the data set, the first 50 terms that occurred most frequently were extracted and their frequencies were transformed to a logarithmic scale. The word cloud in Figure 3.6 is related to pancreatic cancer and to cancer in general. This can be seen by considering synonyms like therapy, dose, survival, metastatic, tumor, etc. The word clouds were generated with the software PyTagCloud[17].

According to the OMIM review about pancreatic cancer[18], three substantially involved genes are KRAS, BRCA2, and CDKN2A. While Figure 3.5 shows their relative frequencies in comparison to other search terms, the timelines in Figure 3.7 illustrate the absolute number of publications per year. The KRAS timeline shows an exponential growth until 2013. The slopes of the BRCA2 and CDKN2A timelines are rather low compared to KRAS, but start much earlier in both plots. There is even a decrease shown for the last three years in the pancreatic cancer data set. One reason for this outcome is the role of KRAS in the regulation of cell proliferation and its higher specificity to pancreatic cancer than in case of BRCA2 and CDKN2A [91]. These examples present approaches to visually inspect the number of publications of specific entities.

One way to investigate the context of selected search terms is to use the PubMed2Go function to generate an HTML page with highlighted entities. The Xapian full text index can be searched with conditional queries for this purpose, e.g. using the Python API. The PubMed2Go documentation describes examples such as a query for the drug R115777[19] in combination with the term pancreatic, excluding the words lung, colon, and ovarian. There is also a query for the drug erlotinib[20] next to the term pancreatic within a range of three words. These drugs are not only applied in case of pancreatic cancer. As already mentioned, it is difficult to cure this type of cancer with what is known so far about therapies using pancreatic cancer-specific drug targets. Therefore, conditional searches can increase the specificity of the search results, focussing on the word neighbourhood and excluding closely related findings.

[16] http://www.ncbi.nlm.nih.gov/books/NBK3827/table/pubmedhelp.T43
[17] https://github.com/atizo/PyTagCloud
[18] http://omim.org/entry/260350?search=%22pancreatic%20cancer%22
[19] http://www.drugbank.ca/drugs/DB04960
[20] http://www.drugbank.ca/drugs/DB00530

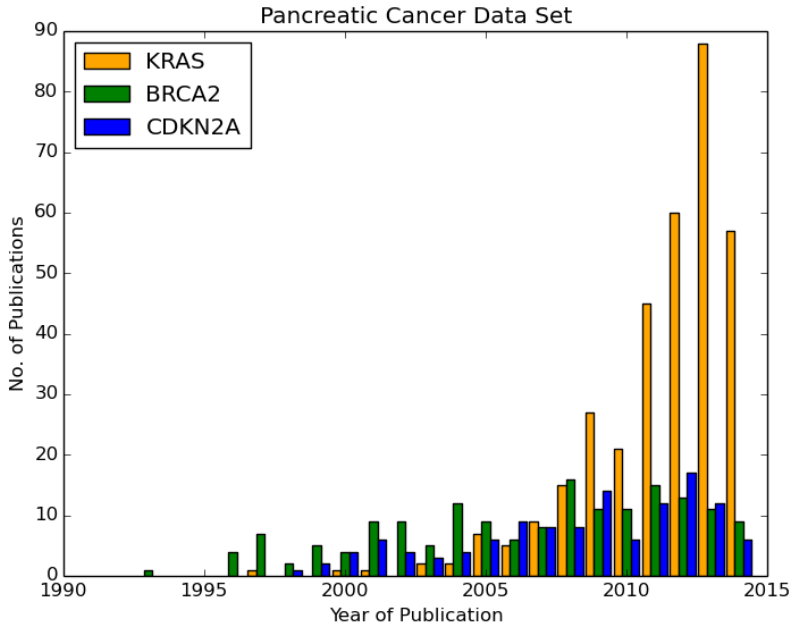


Figure 3.7.: PubMed2Go timelines for the publications of different genes. The Xapian full text index was queried with the gene names KRAS, BRCA2, and CDKN2A. Publication years until 2014 were selected from the PostgreSQL database.

The usage of such HTML pages with highlighted search terms also demonstrates a way to simplify text curation and annotation.

The selective search in Xapian can also be combined with PostgreSQL queries to identify the number of articles published by which authors. Subsequently, their investigated topics can be compared to the entities in Figure 3.5 (described in the GitHub documentation[21]).

Oliver *et al* . showed a rather complex SQL query, selecting the ten journals that published the most articles with the MeSH term "Leukemia" [83]. The slightly modified query to the PubMed2Go database schema pubmed is shown in Table 3.1. The result is displayed in Table 3.2, with the outcome that the order of top range journals did not change a lot within the last ten years. In contrast to the other use cases described here, this selection was applied to the complete PubMed data set. Oliver *et al* . described a difference between warm and cold cache, which is recognised in the PubMed2Go query, too. The query as shown in Table 3.1 took

[21] `https://github.com/KerstenDoering/PubMed2Go/wiki#postgresql-and-xapian`

Table 3.1.: PostgreSQL query to select MeSH term-related journals. The ten journals with the highest number of publications containing the MeSH term "Leukemia" were selected from the complete PubMed data set.

SQL command	
SELECT	mj.medline_ta, count(mj.fk_pmid) as num_of_publications
FROM	pubmed.tbl_medline_journal_info mj
JOIN	pubmed.tbl_mesh_heading msh
ON	mj.fk_pmid = msh.fk_pmid
WHERE	msh.descriptor_name = 'Leukemia'
GROUP BY	mj.medline_ta
ORDER BY	count(mj.fk_pmid) desc
FETCH	first 10 rows only;

almost nine minutes until the result appeared. A resubmission of this query directly afterwards only took half a minute because of a warm cache[22].

Table 3.2.: Results for the query in Table 3.1.

Journal	Number of publications
Blood	1,469
Cancer	748
Leukemia	746
Leuk Res	723
Cancer Res	718
Bone Marrow Transplant	710
Br J Haematol	677
Rinsho Ketsueki	671
Lancet	582
Haematologica	486

Similar to the query in Table 3.1, it can be investigated in which countries the most journals are located, in reference to the pancreatic cancer data set (Figure 3.8). It is shown that the largest number of journals related to pancreatic cancer are located in the United States and England. This does not mean that the authors' institutions are located in these countries.

All examples shown here illustrate approaches how PubMed2Go can be applied to user-specific data sets. For instance, the word cloud in Figure 3.5 and the timeline in Figure 3.7 refer to a manually selected set of search terms, but this can be easily extended to an automatic search pipeline using the proposed BioC named entity

[22] `http://stackoverflow.com/questions/22756092`

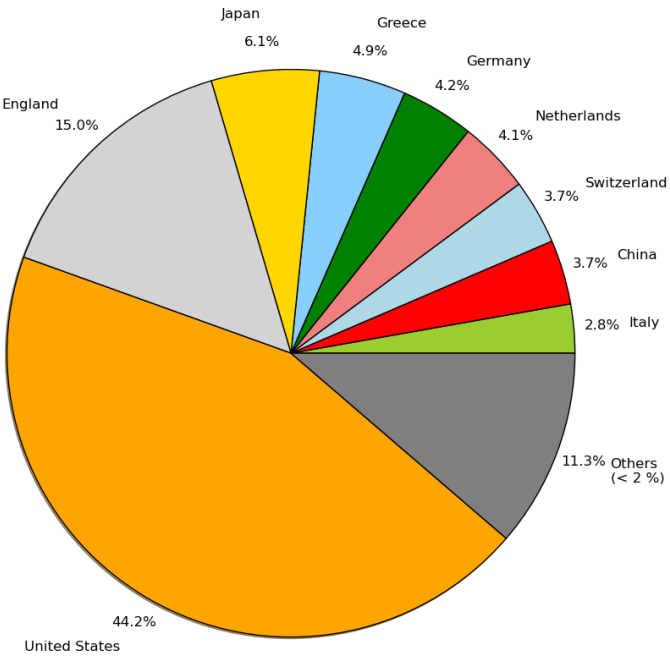


Figure 3.8.: Countries which pancreatic cancer-related journals come from. The percentages were calculated based on a query to the PostgreSQL table pubmed.tbl_medline_journal_info. Fractions below 2 % are summarised as others.

recognition tools. It depends on the user's aim, what kind of data analysis should be included in a text mining application.

3.1.5. Performance

For the complete XML data set with a size of 114 GB, it takes 10.5 days to build the PostgreSQL relational database and another 27 hours to generate the full text index with multiprocessing using a 2 GHz quad-core processor. The time of the indexing process and the size of the index depend on the range of fields in use. A modification of the PubMed2Go scripts, including only abstract titles and texts, but not MeSH terms, keywords, and substances, speeds up this process to 10 hours. The size of the full text index also decreases from 154 GB to 124 GB. It is difficult to compare the runtime to the results from Oliver *et al* . due to different hardware and software system requirements, but increasing computational resources will speed up this process directly by using more CPU cores for multiprocessing.

3.2. Prediction of Functional Compound-Protein Relationships

Parts of this project will be published following this thesis in a similar or identical way.

3.2.1. Related Work

Protein-protein interactions are substantial for cellular processes, involving metabolism, signalling, regulation, and proliferation [9]. Information about interactions can be extracted from databases with experimental and curated data, or identified with text mining approaches [9, 92]. Tikk *et al* . compared nine kernel methods to predict protein-protein interactions in sentences of different text corpora [9]. The best results were achieved by applying the shallow linguistic kernel and the all-paths graph kernel. The all-paths graph kernel represents a deep parsing approach, considering all syntactic connections in a sentence based on a graph structure. Furthermore, the kernel makes use of weighted connections in a so called dependency graph (Section 2.2.4.4). In contrast, the shallow linguistic kernel considers only small word neighbourhoods around the candidate interaction partners. The shallow linguistic kernel was also applied in the area of drug-drug interactions [12]. Other kernel based methods investigated by Tikk *et al* ., e.g. approaches considering only syntax trees, showed an inferior performance.

Small molecules can be metabolised or serve drugs, e.g. inhibiting target proteins [10]. Therefore, compound-protein interactions are essential for the processes in the cell as well, but fewer available approaches exist to characterise them in texts. The search tool for interacting chemicals, abbreviated STITCH, was published in its 4th version in 2013 and connects several information sources of compound-protein interactions. This includes experimental data, e.g. from ChEMBL [93], and data derived from text mining methods, based on co-occurrences and natural language processing [10, 17, 94]. These methods are based on and conntected to the STRING 9.1 database (Search Tool for the Retrieval of Interacting Genes), which contains algorithmic approaches to identify protein-protein interactions [95]. Figure 3.9 shows how STITCH visualises compound-protein interactions, but it can also be used to visualise protein-protein interactions.

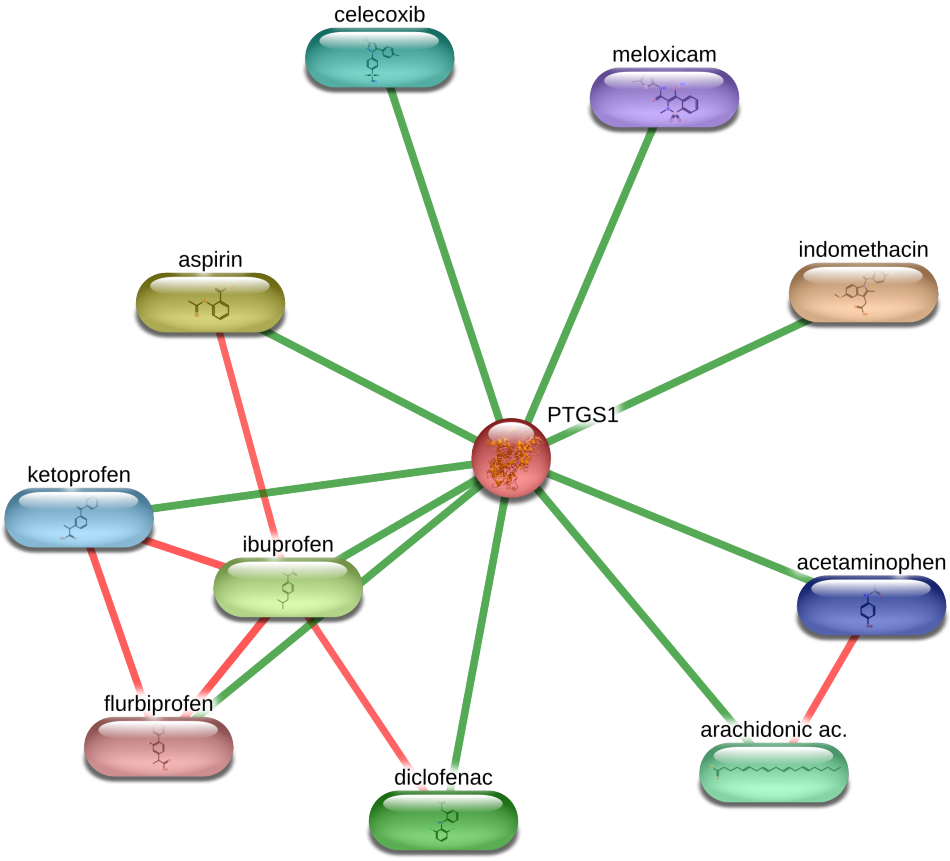


Figure 3.9.: Protein-compound interactions in STITCH [10]. This picture was generated by choosing the example provided by the STITCH start page. The view is centralised for the query protein prostaglandin G/H synthase 1 (PTGS1), also known as Cyclooxygenase-1 (COX-1). The confidence view displays strong associations (high confidence score) with thicker lines. In this case, only the highest confidence scores were selected with a a threshold of 0.9. The view can be changed to include protein-protein interactions as well (blue lines). The red lines show interactions of chemicals. All information sources, including experiments, databases, text mining, etc., are included in this graph.

OntoGene is a text mining Web service for the detection of proteins, genes, drugs, diseases, and chemicals, including their relationships [13]. The relationship identification methods contain on rule-based and machine learning approaches, which were successfully applied in the BioCreative challenges, e.g. in the triage task in 2012[23] [96]. Unfortunately, STITCH and OntoGene do not provide exact statistical measures of their protein-compound interaction prediction methods. Furthermore, no published gold standard corpus of annotated compound-protein interactions could be found for evaluation purposes. In this section, the results of the shallow linguistic and all-paths graph kernel classification will be compared, based on a completely new compound-protein interaction data set. M. Becer was supervised by me to annotate these data sets and to apply the shallow linguistic kernel within his Bachelor Thesis [97]. Furthermore, I supervised E. Abbasian to preprocess textual data for the all-paths graph kernel and to apply the pipeline implemented by Tikk *et al.* during her Master Thesis [47].

3.2.2. Functional Compound-Protein Interactions

The definition of a compound-protein interaction in this thesis includes the direct interaction of a compound and a protein, as shown in Figure 3.10, and indirect functional relations, like the one displayed in Figure 3.11. Furthermore, two types of data sets are distinguished as described in the next two subsections. Data set 1 consists of sentences in which an interaction verb is enclosed by a chemical compound and a protein (Section 2.1.4.3). Data set 2 does not contain such a sentence structure. Every compound-protein pair in a sentence is considered as a potential interaction. If a biomolecule exists as a long form synonym and an abbreviated form in brackets, both terms are considered as individual classification examples.

3.2.2.1. Sentences with Interaction verb

Data set 1 consists of 1,259 sentences. The coloured biomolecules in Figure 3.10, with the interaction verb shown in orange, illustrate a part of an HTML document that was used for compound-protein interaction annotation. Furthermore, this example refers to a direct interaction. In contrast, Figure 3.11 shows an example in which the actual interaction is not inferred by the interaction verb and it illustrates an indirect

[23] `http://www.biocreative.org/tasks/bc-workshop-2012/Triage`

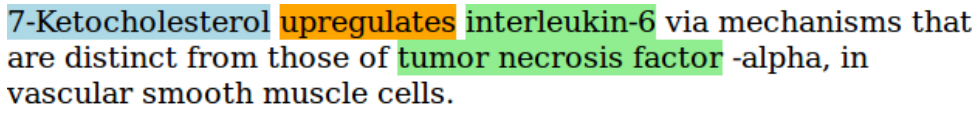


Figure 3.10.: Direct interaction with interaction verb. The orange-coloured verb is enclosed by the PubChem compound 7-ketocholesterol, shown in blue, and the protein interleukin-6, shown in green. The curated status of this sentence is *Interaction*. The protein tumor necrosis factor is annotated as a non-interacting protein in an extra CSV table. This example refers to PubMed ID 18515973, sentence ID 3.

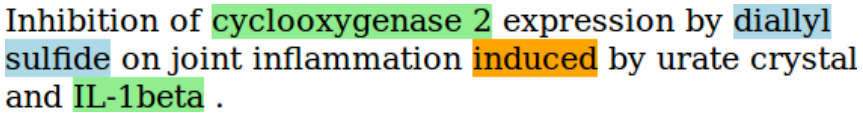


Figure 3.11.: Indirect interaction with interaction verb. The PubChem compound diallyl sulfide and the protein IL-1beta enclose an interaction verb, but these two biomolecules are not in a functional relation. The curated status of this sentence is *Interaction*, because it is stated that diallyl sulfide is influencing cyclooxygenase 2 indirectly by inhibiting its expression. This example refers to PubMed ID 18573688, sentence ID 277.

relationship. Therefore, sentences from data set 1 and their interactions do not necessarily depend on the interaction verb. Both examples contain a non-interacting protein. The status of both sentences is *Interaction* and the non-interacting proteins are annotated in an extra CSV file. Within the preprocessing of data set 1, no part-of-speech tagging was performed. It is possible that interaction words appear which are highlighted as verbs although they are nouns. These cases have been annotated in the CSV file, too. Before annotation of false positive examples, this data set consisted of 2,964 sentences.

3.2.2.2. Sentences without Interaction verb

Data set 2 can be considered analogous to data set 1 with the difference that no interaction verb is allowed to appear between the first and last compound or protein in a sentence. This data set consists of 1,494 sentences. Part of an example sentence is shown in Figure 3.12. Originally, this data set consisted of 5,365 sentences, but it contained many false positive examples.

catalyzed by P450-dependent enzymes, mainly the conversion of cholesterol to pregnenolone by cholesterol side-chain cleavage enzyme (CYP11A).

Figure 3.12.: Direct interaction without interaction verb. This figure shows the second part of a curated sentence in DS2. The small molecule cholesterol is metabolised to pregnenolone by CYP11A. Therefore, this example was annotated with the status *Interaction*. The word conversion can be considered as an interaction word, but it is not a verb. The example refers to PubMed ID 18768916, sentence ID 385.

3.2.2.3. Co-Occurrences

The method of compound-protein co-occurrences can be considered as a prediction method for functional compound-protein relationships (Section 2.1.4.3). Table 3.3 shows the evaluation of this approach.

Table 3.3.: Compound-protein interaction prediction with co-occurrences. Using the concept of co-occurrences as a prediction method means that all pairs of compounds and proteins in a sentence are interaction partners (positive instances). Therefore, the sensitivity is 100 % and the specificity is 0 % by definition. In this case, the precision value equals the accuracy, because there are no true and false negative predictions. In both data sets, the F_1 score is above 70 %, but the interaction verbs in DS1 lead to a higher number of true CPIs. Evaluation results are shown in percent (Sent. - Sentences, Sens. - Sensitivity, Spec. - Specificity, Prec. - Precision, Acc. - Accuracy, F_1 - F_1 score).

DS	# Sent.	# CPIs	# No-CPIs	Sum	Sens.	Spec.	Prec. (Acc.)	F_1
DS1	1259	2042	1264	3306	100	0	61.8	76.4
DS2	1494	1682	1408	3090	100	0	54.4	70.5

3.2.3. Annotation of Functional Relationships

All annotations in data set 1 and data set 2 were assigned manually by working on two large HTML pages with entities highlighted automatically in advance. The colour blue is used for PubChem compounds and entities highlighted in green show UniProt proteins. There are also terms shown in yellow, which refer to GO terms describing cellular processes [14], but these entities were not further analysed within this project. In prolific, GO terms are used as an additional filter criterion in relationship sentences [2]. Data set 1 shows interaction verbs in orange (Section 3.2.2.1). Opening

Index	PubMed-ID	Relationship sentence	Type of interaction	Select	Classify
1	18491395	AFM force-displacement curves were obtained using these modified tips against the surface of a bioactive glass at different stages of an in vitro test performed in a simulated body fluid.	False positive example	False positive example :	Push
2	17604612	Amendment of zinc EDTA and copper EDTA could not suppress the disease significantly when used alone; however, they significantly suppressed the disease in presence of Pf4 -92.	False positive example	False positive example :	Push
3	18515973	7-Ketocholesterol upregulates interleukin-6 via mechanisms that are distinct from those of tumor necrosis factor -alpha, in vascular smooth muscle cells.	Interaction	Interaction :	Push
4	18515973	Among the 7 IL examined, only IL-6 transcript was increased by 7-ketocholesterol treatment in human aorta smooth muscle cells.	Interaction	Interaction :	Push
5	18515973	IL-6 transcripts increased up to 24 h after treatment with 7-ketocholesterol, and this effect was profoundly repressed by treatment with p38 MAPK inhibitors and to a lesser extent JNK inhibitors.	Interaction	Interaction :	Push
6	18515973	7alpha-Hydroxycholesterol, 27-hydroxycholesterol or cholesterol, however, did not induce IL-6 expression.	No Interaction	No interaction :	Push

Figure 3.13.: HTML data set annotation. The interaction verbs are coloured in orange, chemical compounds in blue, and proteins in green. The curated status of each sentence can be assigned as *Interaction*, *No Interaction*, or *False positive example*. Clicking the *Push* button changes the type of interaction.

the HTML page for text annotation the first time shows the status *Pretagged* for each sentence. The curator can select between the three different states *Interaction*, *No Interaction*, or *False positive example* (Figure 3.13), as further explained in the next subsections.

3.2.3.1. Interaction

The non-interacting biomolecules in sentences with direct or indirect functional relationships are annotated in an extra CSV file, as already mentioned in Section 3.2.2.1 and Section 3.2.2.2. Therefore, the CSV file contains the columns non-interacting compound (NI-C) and non-interacting protein (NI-P). Nevertheless, there is the case in which the sentence contains interactions, but the non-interacting partners cannot be determined by the algorithm. If there are two interacting pairs $C_1 - P_1$ and $C_2 - P_2$ in the same sentence, but C_1 is not interacting with P_2 and C_2 not with P_1, the parser is not able to find the correct relationships with the annotations in the CSV file. The reason is that a biomolecule cannot be entered as non-interacting and actually interact the same time in one sentence. Therefore, such a sentence has to receive the status *False positive example*. However, this is a very rare case.

3.2.3.2. No Interaction

The examples in Section 3.2.2.1 and Section 3.2.2.2 show non-interacting biomolecules, because nothing is stated about a relationship. The other case would be the explicit

use of a word like *no* or *did not* as shown in Figure 3.14.

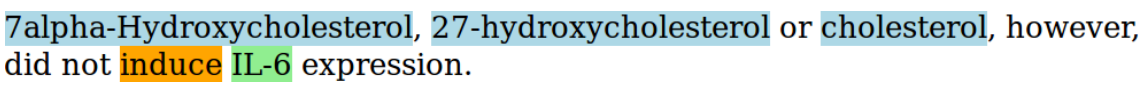


Figure 3.14.: No interaction. This figure shows a sentence from data set 1 in which the status *No Interaction* is induced by the terms *did not*. This example also belongs to the abstract with the PubMed ID 18515973, like the one given in Figure 3.10. It has got the sentence ID 6, as shown in Figure 3.13.

3.2.3.3. False Positive Example

False positive examples can occur in many sentences, but only if there are no interacting or non-interacting biomolecules left, the status remains *False positive example*. Otherwise, the molecule will be added to the columns false positive compound (FP-C) or false positive protein (FP-P) in the CSV file to be ignored by the classifier. In the example shown in Figure 3.15, two false positive examples can be seen. Dopamine is not a compound in this sentence, because it is part of the protein dopamine transporter. In this case, the word transporter is added to an extra blacklist to improve the named entity recognition algorithm that currently annotates compounds in a sense of wrong word neighbourhoods. PCR is a false positive example, because it refers to the polymerase chain reaction. This word was identified by Whatizit. Therefore, only a blacklist of typical false positive abbreviations can improve the quality or alternatively, using another algorithm.

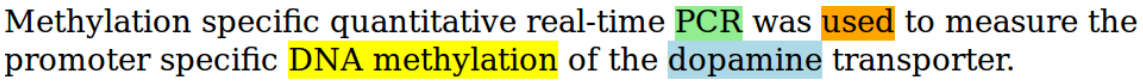


Figure 3.15.: False positive example. PCR is not a protein, because it abbreviates the technology polymerase chain reaction. Dopamine is highlighted as a compound, but it is part of the transporter protein. Therefore, this is a false positive example, too. DNA methylation is highlighted in yellow, because it was recognised as a GO term by Whatizit. The example refers to PubMed ID 18768916, sentence ID 385.

If a compound or a protein consists of multiple terms, but the parts of the synonym were recognised as different entities, the words will be considered as false positive hits, too. Compounds and proteins that were not recognised as biomolecules at all, are not further considered in this section.

3.2.4. Generation of Data Sets

Usually compounds are referred to as small molecules up to a molecular weight of 1,000 kDa for which a synonym and a structure is contained in PubChem [3]. Similarly, gene and protein names refer to UniProt IDs, to be able to map a synonym to its protein sequence from UniProt [4]. PubChem synonyms were annotated with the approach used in the Web services CIL [1] and prolific [2], applying the Hettne-Rules [49]. Proteins were annotated using the Web service Whatizit [7] also used in CIL and prolific.

Table 3.4.: PostgreSQL query to select all PubMed IDs from 2009 ascending. The column pub_date_year was queried with the publication year 2009 and fk_pmid with the PubMed ID (no ";" character). This query was sent to the PostgreSQL database phabidb, which contains all PubMed articles until January 2015, based on the PubMed2Go framework.

```
\copy (
SELECT      fk_pmid
FROM        pubmed.tbl_journal
WHERE       pub_date_year = 2009
ORDER BY    fk_pmid ASC
) to        'pmids_2009.csv' delimiter ','
```

The two data sets were generated by selecting the first 40,000 abstracts from PubMed, ordered by PubMed ID from the prolific PostgreSQL database and based on the PubMed2Go framework. The SQL command is shown in Table 3.4. The first 20,000 publications were used for data set 1 and the second 20,000 for data set 2. The MongoDB (Section 2.1.2) of the Web service prolific was queried for all existing compound-protein pairs for the given PubMed IDs with a NoSQL command like in Table 3.5. The result is shown in Table 3.6. The selected UniProt ID and the PubChem parent compound ID in this example query refer to Figure 3.10. Similarly, a MongoDB query for data set 2 would contain the condition 'verb_metric': false. All IDs for these selections are generally stored in the code of each HTML page to be used in ongoing preprocessing steps after text annotation.

The sentences in the HTML files contain misplaced spaces, e.g. in case of words in round brackets. The reason is that the sentences are extracted from preprocessed abstracts and titles that were generated with a parser for Whatizit by another developer. For the new data set used in the pipeline of the all-paths graph kernel (Figure 3.23 in Section 3.2.6), most of these characters could be properly arranged

Table 3.5.: NoSQL query to select database entries from MongoDB. With this syntax, all available fields will be shown that match the criteria of containing the PubMed ID 18515973, the PubChem parent compound ID 91474, and the UniProt ID P41693 in a sentence with a verb enclosed. This is ensured by the boolean statement 'verb_metric': true.

```
db.docs.find(
{'pmid':          {'$exists' : true},
'cid':            {'$exists' : true,'$ne':''},
'pcid':           {'$exists':true},
'pid':            {'$exists' : true,'$ne':''},
'verb_metric':    true,
'pmid':           18515973,
'pid':            'P41693',
'pcid':           91474})
```

Table 3.6.: Results for NoSQL query in Table 3.5. As no limitation of selected fields to display is given, part of the queried fields will be shown, too. It can be seen, that the PubChem compound ID and the parent ID are identical and that this compound-protein pair appears with several other interaction verbs in the same abstract. Every database entry contains an object ID that is provided by the MongoDB engine itself while insertion of new data.

```
{ '_id' :                                  ObjectId('5285f189735df7f69d9c100d'),
'cid':                                     91474,
'pcid':                                    91474,
'pid':                                     P41693,
'sentence_metric' : true 'verb_metric':    true,
'pmid':                                    NumberLong(18515973),
'pid':                                     'P41693',
'verbs':                                   ['enhanced', 'impaired',
                                           'increased', 'release', 'upregulates'] }
```

with regular expressions. If a sentence contained a highlighted protein, an algorithm analysed whether there is also a PubChem compound synonym. In this case, the parent compound ID was inserted into a surrounding XML tag. If there was a verb, it was highlighted, too. The algorithm did not insert a compound tag if a protein tag already appeared around the selected position and always inserted the XML tag around the longest matching compound name. Therefore, no nested or overlapping XML tags were produced, except for some GO terms, because these XML tags were excluded later anyway. Every new sentence received a sentence ID such that multiple sentences in an abstract could be considered as individual instances to classify. For

the HTML page, XML tags were replaced with mark tags to highlight the entities in different colours. Furthermore, a simple JavaScript button was implemented to store the status of each sentence directly in the HTML page (Figure 3.13). After all sentences were curated, the HTML document could be parsed to generate the classifier-specific input format as described in the next sections. The prerequisites for the shallow linguistic kernel and the all-paths graph kernel approach are different such that the workflow to process the curated data sets also differ.

3.2.5. Shallow Linguistic Kernel Pipeline

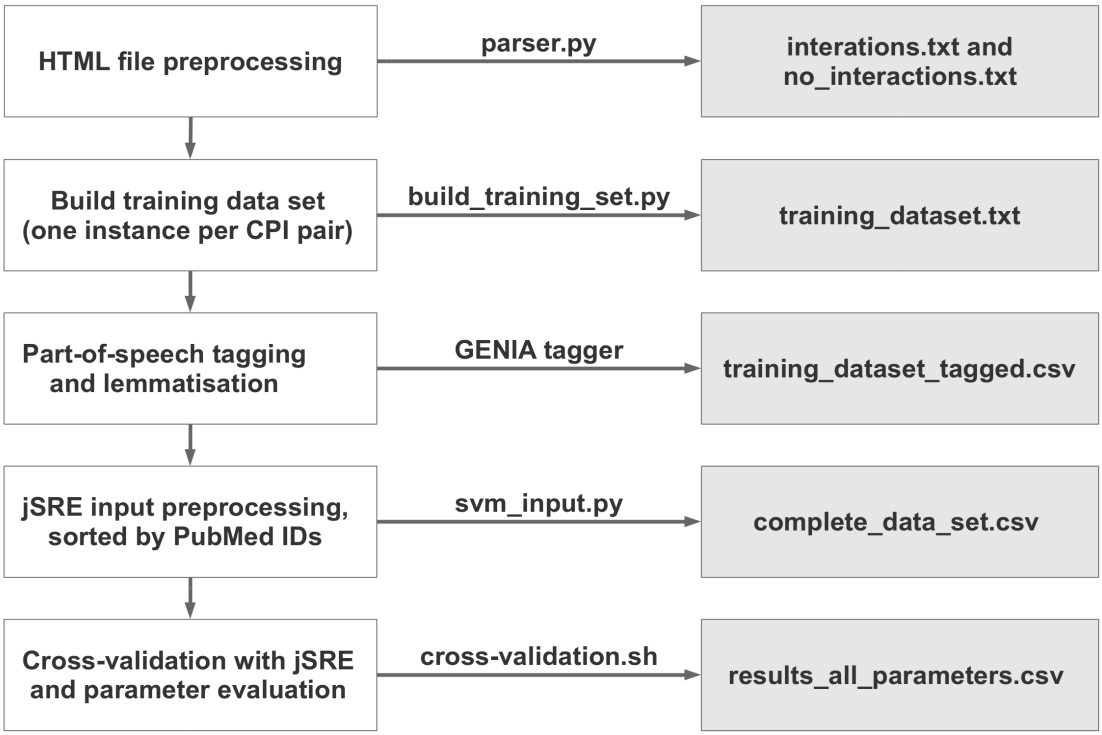


Figure 3.16.: Shallow linguistic kernel workflow. This workflow shows which steps are involved in preprocessing the data for the jSRE software, running it with a parameter selection, and evaluating the 10-fold cross-validation results. The workflow starts with processing the HTML file and thereby considering the annotations in the extra CSV file. The shallow linguistic kernel can be applied with different values for window size and n-gram (Section 3.2.5). The cross-validation runs are averaged and summarised for the different shallow linguistic kernel parameters in the file results_all_parameters.csv.

The workflow in Figure 3.16 illustrates the basic steps in the application of the

shallow linguistic kernel to data set 1 and data set 2 with the software jSRE[24] (java Simple Relation Extraction) [63], which will be explained within the next subsections in more detail.

3.2.5.1. Preprocessing of the Curated Data Set

The script parser.py extracted sentences with the status *Interaction* or *No Interaction*. All sentences with the status *False positive example* were ignored. For each of both data sets one file with interaction sentences and without interaction sentences was generated. An example is shown in Figure 3.17.

```
18515973-3  <mark style="background-color:lightblue;" id="91474">7-
Ketocholesterol</mark> <mark style="background-color:orange;">upregulates</mark>
<mark style="background-color:lightgreen;" id="P41693">interleukin-6</mark> via
mechanisms that are distinct from those of <mark style="background-
color:lightgreen;" id="Q8JFG3">tumor necrosis factor</mark> -alpha, in vascular
smooth muscle cells.
```

Figure 3.17.: Extraction of HTML sentences. This sentence refers to the example given in Figure 3.10 and shows how the coloured HTML sentence is encoded with mark tags and biomolecule IDs from PubChem and UniProt.

3.2.5.2. Generation of Training Instances

The script build_training_set.py processes interactions.txt and no_interactions.txt in a way that it generates one new instance of a sentence for each interacting or non-interacting compound-protein pair in a sentence. Therefore, an incremental index is needed for every pair in a sentence. Furthermore, this step involves excluding all remaining false positive examples by parsing the annotations in the extra CSV file. Figure 3.18 shows that the mark tags from Figure 3.17 have been converted to the jSRE format. This format consists of a token ID, the token or term itself, its lemmatised form or base form, a part-of-speech tag, the entity type, and whether it is the target T or the agent A, as explained in the jSRE user guide. All other terms receive an O for other entity. As all relationships are undirected, the entity type *Comp* for compound is always the agent and the type *Prot* (for protein) the target. The elements of the jSRE format are separated by the doubled character '&' and each leading token ID is separated by a space character. These token IDs are added to

[24] `https://hlt-nlp.fbk.eu/technologies/jsre`

the already processed biomolecule synonyms after the GENIA tagger preprocessing steps of all other sentence elements, as described in the next subsection.

Figure 3.18 displays the part-of-speech tags NN and NNP. The difference is that NNP refers to a proper noun. Such terms are special, because they start with a capital letter. This example sentence also contains an interaction pair which means that a second instance is contained in training_dataset.txt that is marked with class 1 and in which the target is changed to interleukin-6.

```
0   18515973-3-0     7-Ketocholesterol&&7-Ketocholesterol&&NNP&&Comp&&A
upregulates   interleukin-6&&interleukin-6&&NN&&Prot&&O  via mechanisms that are
distinct from those of
tumor_necrosis_factor&&tumor_necrosis_factor&&NN&&Prot&&T  -alpha , in vascular
smooth muscle cells .
```

Figure 3.18.: Extraction of HTML sentences. This sentence shows a non-interacting pair as indicated by the number zero in the beginning of this training instance. The target is the protein tumor necrosis factor and the agent the compound 7-ketocholesterol as indicated by the capital letter T and A.

3.2.5.3. Lemmatisation and Tokenisation

The GENIA tagger is used to generate the base or dictionary form of all terms, referring to the process of lemmatisation. The script build_data_set.py inserts

```
Inhibition      Inhibition      NN      B-NP    O
of              of              IN      B-PP    O
NF-kappaB       NF-kappaB       NN      B-NP    B-protein
activation      activation      NN      I-NP    O
reversed        reverse         VBD     B-VP    O
the             the             DT      B-NP    O
anti-apoptotic  anti-apoptotic  JJ      I-NP    O
effect          effect          NN      I-NP    O
of              of              IN      B-PP    O
isochamaejasmin isochamaejasmin NN      B-NP    O
.               .               .       O       O
```

Figure 3.19.: GENIA tagger. This example from the GENIA homepage shows from left to right for each term in the sentence the tokenised words, lemmatised words, part-of-speech tags, text chunks, and identified proteins. Chunks are semantically connected text fragments (Section 2.2.4.4), e.g. noun phrases (NP) like Inhibition, NF-kappaB activation, the anti-apoptotic effect, and isochamaejasmin. Each character 'B' represents the start of a new chunk. Other phrases are PP (prepositional phrase) and VP (verb phrase). IN respresents a preposition, JJ and adjective, DT a singular determiner, and VBD a verb in past tense.

spaces between punctuation signs and the rest of the terms such that brackets for

abbreviations or phrases are already tokenised. Normally, the GENIA tagger uses its own tokeniser[25], but it can be called with the paramter '-nt' to perform whitespace tokenisation, which was done in this case. Furthermore, so called chunk tags are generated by the GENIA tagger [98], e.g. to recognise noun phrases (Figure 3.19[26]). However, part of the tokens in a sentence possibly will not be tagged as part of a chunk. This approach is used in several information extraction systems [99].

```
0   18515973-11-0   0&&IL-6&&IL-6&&NNP&&Prot&&O 1&&release&&release&&NN&&O&&O
2&&by&&by&&IN&&O&&O 3&&7-ketocholesterol&&7-ketocholesterol&&NN&&Comp&&A
4&&although&&although&&IN&&O&&O 5&&significant&&significant&&JJ&&O&&O
6&&was&&be&&VBD&&O&&O 7&&as&&as&&RB&&O&&O 8&&remarkable&&remarkable&&JJ&&O&&O
9&&as&&as&&IN&&O&&O 10&&that&&that&&DT&&O&&O 11&&induced&&induce&&VBN&&O&&O
12&&by&&by&&IN&&O&&O 13&&TNF-alpha&&TNF-alpha&&NNP&&Prot&&T
```

Figure 3.20.: Sentence in jSRE format with GENIA chunks and without punctuation. This example refers to the sentence ID 11 from the abstract with the PubMed ID 18515973. Only words inside chunks, as generated by the GENIA tagger, were included to this training instance.

```
0   18515973-11-0   0&&IL-6&&IL-6&&NNP&&Prot&&O 1&&release&&release&&NN&&O&&O
2&&by&&by&&IN&&O&&O 3&&7-ketocholesterol&&7-ketocholesterol&&NN&&Comp&&A
4&&,&&,&&,&&O&&O 5&&although&&although&&IN&&O&&O
6&&significant&&significant&&JJ&&O&&O 7&&,&&,&&,&&O&&O 8&&was&&be&&VBD&&O&&O
9&&not&&not&&RB&&O&&O 10&&as&&as&&RB&&O&&O 11&&remarkable&&remarkable&&JJ&&O&&O
12&&as&&as&&IN&&O&&O 13&&that&&that&&DT&&O&&O 14&&induced&&induce&&VBN&&O&&O
15&&by&&by&&IN&&O&&O 16&&TNF-alpha&&TNF-alpha&&NNP&&Prot&&T 17&&.&&.&&.&&O&&O
```

Figure 3.21.: Complete sentence in jSRE format. This sentence shows a similar structure as in Figure 3.20, but the words were not reduced to GENIA chunks and punctuation is included. The word "not" contains the part-of-speech tag RB, which represents an adverb.

The jSRE pipeline was first tested only with words inside a chunk excluding punctuation and then with all terms including punctuation. The outcome differs and is described in the next subsection. Depending on the inclusion of all words or just parts of chunks, the concatenated tokens result in a different sentence or phrase structure in complete_data_set.csv as shown in Figure 3.20 and Figure 3.21 (generated with the script svm_input.py). The complete sentence displayed in these figures is "IL-6 release by 7-ketocholesterol, although significant, was not as remarkable as that induced by TNF-alpha." Comparing the two figures, the word "not" and the three punctuation signs are not included in the chunking model.

[25] http://www.cis.upenn.edu/~treebank/tokenizer.sed
[26] http://www.nactem.ac.uk/tsujii/GENIA/tagger

3.2.5.4. Results

The total runtime for the workflow as illustrated in Figure 3.16 on data set 1 and data set 2 is 1.25 h for the version using chunk tags from the GENIA tagger and without punctuation as well as for the version using all words of a sentence and punctuation. The size of the complete development folder is also quite similar. The version with more words is around 7 MB larger, resulting in 490 MB.

All parameter combinations in the range 1-3 for window size and n-gram were evaluated (Section 3.2.5). For both data sets, with and without using chunks, the parameter selection window size 3 and n-gram 3 shows the highest accuracy and highest F_1 score. Therefore, this selection will be focussed in the comparison of the jSRE results. Nevertheless, the results within the models are close to each other. For all models shown here, it can be seen that a lower value of n-gram leads to a higher specificity and a lower recall.

In all of the following four tables, the best two results per column are marked in bold letters. The first number in the first column refers to the parameter window size, the second one represents the parameter n-gram (Section 3.2.5). Results are shown in percent.

Considering data set 1, the version without chunking shows a very similar, but slightly lower F_1 score with an around 3 % lower recall, but an around 2 % higher precision. Although the accuracy is only slightly better in the version without chunking, its specificity is clearly higher with a difference of 7.4 %. This can be seen by comparing Table 3.7 and Table 3.8.

Table 3.7.: Data set 1 results with chunk tags and without punctuation.

Parameter	Sensitivity	Specificity	Precision	Accuracy	F_1 score
11	75.3	**51.9**	71.9	66.4	73.4
12	79.3	49.4	72.0	67.9	75.4
13	**81.9**	47.9	72.0	**68.9**	**76.5**
21	76.1	51.7	72.0	66.8	73.9
22	79.9	48.5	71.8	67.9	75.5
23	**82.2**	47.1	71.8	68.8	**76.5**
31	74.5	**53.1**	72.2	66.3	73.1
32	78.9	50.6	**72.4**	68.1	75.4
33	**81.9**	48.7	**72.3**	**69.2**	**76.7**

Table 3.8.: Data set 1 results including all words of a sentence and punctuation.

Parameter	Sensitivity	Specificity	Precision	Accuracy	F_1 score
11	75.1	58.7	**75.0**	68.8	74.9
12	77.0	56.5	74.4	69.2	75.5
13	**78.0**	55.2	74.0	69.3	75.8
21	75.0	**58.8**	74.9	68.8	74.7
22	76.5	57.2	74.5	69.2	75.4
23	77.9	56.7	74.6	**69.8**	**76.1**
31	75.2	**59.0**	**75.1**	69.0	75.0
32	76.8	57.4	74.7	69.4	75.6
33	**78.6**	56.1	74.5	**70.0**	**76.4**

Table 3.9.: Data set 2 results with chunk tags and without punctuation.

Parameter	Sensitivity	Specificity	Precision	Accuracy	F_1 score
11	76.3	66.5	73.3	71.8	74.6
12	80.2	65.0	73.5	73.3	76.6
13	**82.1**	62.1	72.3	73.0	**76.8**
21	75.2	67.5	73.6	71.7	74.2
22	78.9	66.2	73.8	73.1	76.1
23	**80.9**	63.7	72.9	73.1	76.5
31	75.1	**68.6**	**74.3**	72.1	74.5
32	78.9	**67.3**	**74.5**	**73.6**	76.5
33	80.7	65.7	73.9	**73.8**	**77.0**

Table 3.10.: Data set 2 results including all words of a sentence and punctuation.

Parameter	Sensitivity	Specificity	Precision	Accuracy	F_1 score
11	78.5	71.2	76.6	75.2	77.4
12	79.6	70.9	76.7	75.7	78.0
13	**81.5**	68.9	75.9	75.7	78.5
21	78.2	72.1	77.1	75.4	77.6
22	79.1	**73.2**	**78.0**	76.4	78.5
23	80.8	71.6	77.4	**76.6**	**79.0**
31	78.1	72.2	77.2	75.4	77.5
32	79.0	**72.4**	**77.6**	76.0	78.2
33	**81.4**	71.2	77.3	**76.8**	**79.2**

In case of data set 2, all results in the model without chunking in Table 3.10 are better than the values shown in Table 3.9. The F_1 score is around 2 % better because

of a 0.7 % higher sensitivity and a 3.4 % higher precision. The accuracy is 3.0 % better because of the 5.5 % higher specificity.

The inclusion of all words in a sentence and the punctuation increased the values of specificity and precision. While the sensitivity was slightly better in case of data set 2, it decreased by 3.3 % in the model of data set 1. In general, the jSRE performance on data set 2 reaches a remarkably higher specificity and a slightly better value of precision. The recall of data set 1 and data set 2 shows similar results.

3.2.6. All-Paths Graph Kernel Pipeline

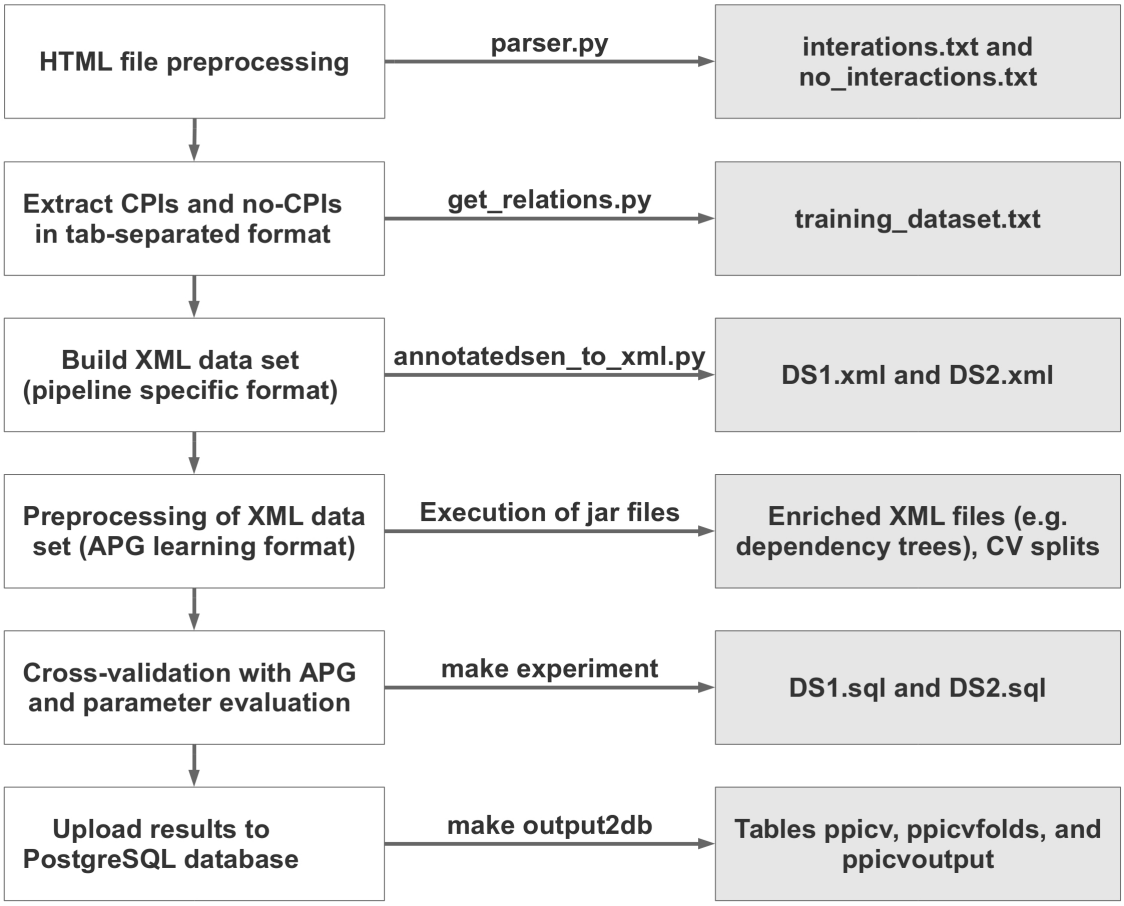


Figure 3.22.: All-paths graph kernel workflow. After extracting all annotated sentences in a tab separated format with the script get_relations.py, several XML preprocessing steps are needed to generate the all-paths graph kernel learning format [9]. The 10-fold cross-validation results are generated, transformed into SQL format, and uploaded to a PosgreSQL database by using a makefile.

The first step in the workflow shown in Figure 3.22 is the same as in Figure 3.16 (shallow linguistic kernel workflow) and refers to the extraction of annotated sentences from the HTML files for data set 1 and data set 2. The script get_relations.py is similar to the shallow linguistic kernel implementation build_training_set.py described in Section 3.2.5.2, but it does not have to generate the '&&'-structure, which is required for the jSRE software. Instead, it directly exports a CSV file with tab-separated interaction and no-interaction pairs as shown in Figure 3.23. The complete sentence displayed in this figure is "7-Ketocholesterol also enhanced IL-6 release from VSMC."

```
18515973-10 <compound-id="91474">7-Ketocholesterol</compound-id> also enhanced <protein-
id="O35736,P05231,P08505,P20607,P26892,P26893,P29455,P41323,P41683,P41693,P46650,P51494,
protein-id> release from VSMC.  7-Ketocholesterol__IL-6__interaction
```

Figure 3.23.: Sentence with tab-separated functional interaction pairs. This example refers to PubMed ID 18515973 (sentence ID 10) from data set 1. As shown in Figure 3.10, the compound 7-ketocholesterol is interacting with the protein IL-6. The biomolecules and the status itself are connected with two underscores and every compound-protein interaction pair is separated with a tab character. The list of UniProt IDs is not fully displayed here. VSMC is the abbreviation for vascular smooth muscle cells.

The generation of the all-paths graph kernel learning format will be performed by executing several jar files. The input for these conversion steps is an XML file which is built by using annotatesen_xml.py. The format of this XML file is shown in Figure 3.24. While this XML format is used by all kernels in the protein-protein

```
−<document id="DS1.d279" origId="18515973">
 −<sentence id="DS1.d279.s0" origId="18515973-10" text="7-Ketocholesterol also enhanced IL-6
  release from VSMC.">
    <entity id="DS1.d279.s0.e0" origId="91474" charOffset="0-17" type="compound" text="7-
    Ketocholesterol"/>
    <entity id="DS1.d279.s0.e1"
    origId="O35736,P05231,P08505,P20607,P26892,P26893,P29455,P41323,P41683,P41693,P4665
    charOffset="32-36" type="protein" text="IL-6"/>
    <pair e1="DS1.d279.s0.e0" e2="DS1.d279.s0.e1" id="DS1.d279.s0.i0" interaction="True"/>
  </sentence>
```

Figure 3.24.: XML format before preprocessing. This sentence refers to the example shown in Figure 3.23 and shows the XML structure of documents with sentences that have compound and protein entities containing a boolean interaction status. As in Figure 3.23, the list of UniProt IDs is not fully displayed here.

interaction software package from Tikk *et al* . [9], the all-paths graph kernel learning format needs an XML file enriched with a dependency tree for each sentence as explained in the next subsection.

3.2.6.1. Preprocessing

The sentences have to be tokenised at first and every token needs a part-of-speech tag, a token ID, and a character offset starting at zero for every sentence. This is done by the Charniak-Lease tokeniser as shown in Figure 3.25.


```
- <tokenization tokenizer="Charniak-Lease">
    <token POS="NNP" charOffset="0-16" id="t_1" text="7-Ketocholesterol"/>
    <token POS="RB" charOffset="18-21" id="t_2" text="also"/>
    <token POS="VBD" charOffset="23-30" id="t_3" text="enhanced"/>
    <token POS="JJ" charOffset="32-35" id="t_4" text="IL-6"/>
    <token POS="NN" charOffset="37-43" id="t_5" text="release"/>
    <token POS="IN" charOffset="45-48" id="t_6" text="from"/>
    <token POS="NNP" charOffset="50-53" id="t_7" text="VSMC"/>
    <token POS="." charOffset="54-54" id="t_8" text="."/>
  </tokenization>
```


Figure 3.25.: Tokenisation. This sentence refers to the example shown in Figure 3.23 and shows the Charniak-Lease tokenisation with part-of-speech tags.

Afterwards, a syntax tree can be build by the Charniak-Johnson-McClosky parser[27] (Section 2.2.4.4) [100]. The structure is determined by brackets around the terms and their part-of-speech tags as shown in Figure 3.26. The character offsets of the original text are aligned with the new offsets of the sentence after bracketing.


```
- <bracketing tokenizer="Charniak-Lease" parser="Charniak-Johnson-McClosky" bracketing="(S1
  (S (NP (NNP 7-Ketocholesterol)) (ADVP (RB also)) (VP (VBD enhanced) (NP (JJ IL-6) (NN release))
  (PP (IN from) (NP (NNP VSMC)))) (. .)))">
    <charOffsetMapEntry sentenceTextCharOffset="0-16" bracketingCharOffset="16-32"/>
    <charOffsetMapEntry sentenceTextCharOffset="18-21" bracketingCharOffset="46-49"/>
    <charOffsetMapEntry sentenceTextCharOffset="23-30" bracketingCharOffset="62-69"/>
    <charOffsetMapEntry sentenceTextCharOffset="32-35" bracketingCharOffset="80-83"/>
    <charOffsetMapEntry sentenceTextCharOffset="37-43" bracketingCharOffset="90-96"/>
    <charOffsetMapEntry sentenceTextCharOffset="45-48" bracketingCharOffset="108-111"/>
    <charOffsetMapEntry sentenceTextCharOffset="50-53" bracketingCharOffset="123-126"/>
    <charOffsetMapEntry sentenceTextCharOffset="54-54" bracketingCharOffset="135-135"/>
  </bracketing>
```


Figure 3.26.: Syntactic tree parse. The tree structure can be read by following every pair of brackets separated by a space with a leading part-of-speech tag from left to right.

The syntax tree parse in Figure 3.26 can be converted to a dependency tree (Section 2.2.4.4) with the Stanford conversion tool[28]. The result in Figure 3.27 shows that every token is connected to another one with a directed description of their type of dependency.

[27] https://github.com/BLLIP/bllip-parser
[28] http://nlp.stanford.edu/software/lex-parser.shtml

```
−<parse tokenizer="Charniak-Lease" parser="Charniak-Johnson-McClosky">
    <dependency id="d_1" t1="t_3" t2="t_1" type="nsubj" origId="nsubj(enhanced-3,
    7-Ketocholesterol-1)"/>
    <dependency id="d_2" t1="t_3" t2="t_2" type="advmod" origId="advmod(enhanced-3,
    also-2)"/>
    <dependency id="d_3" t1="t_5" t2="t_4" type="amod" origId="amod(release-5, IL-6-4)"/>
    <dependency id="d_4" t1="t_3" t2="t_5" type="dobj" origId="dobj(enhanced-3, release-5)"/>
    <dependency id="d_5" t1="t_3" t2="t_7" type="prep_from" origId="prep_from(enhanced-3,
    VSMC-7)"/>
  </parse>
```

Figure 3.27.: Dependency tree parse. This tree structure shows a directed dependency between every pair of tokens, built with the Charniak-Johnson-McClosky parser.

Putting together all these XML elements with the original sentence shown in Figure 3.24, the presented sentence is completely transformed to the all-paths graph learning format. Finally, the number of documents is randomly split to 10 equally sized parts. As the number of sentences per document differs, these parts do not have to be absolutely equal in size. The packages are zipped and copied to the appropriate folders in the main project directory such that the parameter evaluation can start. All these steps including the final part in Figure 3.22 are automatically executed with a shell script. The software package from Tikk *et al* . has got a complex structure and a detailed PDF[29] describing how to perform the different steps for each kernel. Therefore, many folders and files from this package could be removed as they were not directly needed for the all-paths graph kernel pipeline. Furthermore, some special configurations and debugging steps were performed to isolate the mandatory preprocessing steps for the all-paths graph kernel learning format and to reduce the unzipped size of the main folder from 107.1 MB to 27.3 MB.

3.2.6.2. Results

The two main make commands shown in Figure 3.22 as well as the single cross-validation results can be found in the Tikk et al. software documentation. The first make command runs the steps of training the sparse regularized least squares (RLS) model for every cross-validation after generating a linearised feature representation and normalising the data. Four chosen values of the regularisation parameter 'c' are tested, namely 0.25, 0.50, 1.00, and 2.00 (Section 2.2.4.5). The second make

[29] `https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/ppi-benchmark/ppi-benchmark-tar.gz`

command inserts the prediction results for every compound-protein pair and every parameter selection into the PostgreSQL table ppicvoutput. Another table ppicvfolds determines an ID for every corpus-parameter combination which is inserted to the PostgreSQL table ppicv. Considering one corpus with four regularisation parameter values and 10 cross-validation runs, 40 rows will be inserted into the PostgreSQL table ppicv. The average for every 10 rows for data set 1 and data set 2 are given in Table 3.11 and Table 3.12. The runtime for each of the two preprocessing steps was between 30 and 60 min. The main calculation with the first make command lasted around 4.5 h for data set 1 and around 4 h for data set 2.

In the following two tables, the best two results per column are marked in bold letters. The first number in the first column refers to the sparse RLS regularization parameter 'c'. Results are shown in percent.

Table 3.11.: Data set 1 results for the all-paths graph kernel pipeline.

Parameter	Sensitivity	Specificity	Precision	Accuracy	F_1 score	AUC
0.25	82.6	**56.3**	76.2	73.1	79.1	79.1
0.50	82.7	**56.8**	**76.8**	**73.5**	79.4	**79.3**
1.00	**86.0**	50.9	74.9	73.3	**80.0**	**79.3**
2.00	**84.2**	**56.3**	**76.7**	**74.2**	**80.2**	79.0

Table 3.12.: Data set 2 results for the all-paths graph kernel pipeline.

Parameter	Sensitivity	Specificity	Precision	Accuracy	F_1 score	AUC
0.25	78.7	**71.2**	**77.7**	**75.5**	**78.0**	**82.0**
0.50	78.0	**72.8**	**78.5**	**75.6**	**78.0**	**82.0**
1.00	**78.6**	70.5	77.5	75.0	77.8	81.8
2.00	**79.5**	69.4	77.0	75.0	**78.0**	81.5

The specificity is generally higher for a lower value of 'c' while the sensitivity shows better values for a larger regularization parameter. The value 2.00 showed the best F_1 score on data set 1, although the difference to the result for value 1.00 is only 0.2 %. In case of data set 2, the values 0.25, 0.50 and 2.00 showed the same F_1 score, but specificity, precision, and recall were slightly higher for the first two parameter selections. The specificity on data set 2 was around 15 to 20 % better than on data set 1 and precision as well as accuracy were also slightly higher than in data set 1. The F_1 score on data set 1 was 1 to 2 % better because of the 4 to 8 % higher sensitivity. As all results in data set 2 are close to each other, the best overall result

is shown for the regularisation parameter value 2.00. However, the AUC values for data set 1 are all around 79 % and in data set 2 around 82 %.

3.2.7. Summary

The jSRE implementation reaches the same F_1 score on data set 1 as the co-occurrences approach, but with much better specificity. The precision is also higher and the recall is in a good range of around 80 %. Comparing the shallow linguistic kernel results on data set 2 with the co-occurrences, the jSRE implementation reaches an around 9 % better F_1 score of 79.2 % in the model without chunks.
Considering the all-paths graph kernel implementation and the co-occurrences results, the all-paths graph kernel models reach a 4 % better F_1 score in case of data set 1 and 8 % better F_1 score in case of data set 2. Therefore, the best all-paths graph kernel runs perform better than the jSRE models in terms of the F_1 score on the data set with interaction verbs, but slightly worse on data set 2. For data set 2, this is only true for the shallow linguistic kernel model without chunking. In general, the shallow linguistic kernel model with chunking on data set 2 shows slightly lower values. Although the all-paths graph kernel model outperforms the jSRE implementation without chunking on data set 1, the shallow linguistic kernel model shows around the same specificity in case of the model with the best F_1 score and even better specificity values for the parameter selections with a slightly lower F_1 score (n-gram value 1).
To summarise, the jSRE implementation shows better results on data set 2 using the model without chunks and the all-paths graph kernel model shows better results on data set 2 in all cases. The runtime of the all-paths graph kernel implementations is up to four times slower than the jSRE pipeline.

3.3. Toxicity Prediction

This work has not yet been published in a journal.

3.3.1. Related Work

Many freely available and commercial tools exist that perform toxicity prediction in terms of physico-chemical properties, biological effects, and toxicological endpoints in an organism [5, 101]. Toxicological endpoints are acute oral toxicity, genotoxicity, carcinogenicity, reproductive toxicity, hepatotoxicity, neurotoxicity, or cytotoxicity [5]. Software examples are Toxtree[30] and Lazar[31] (freely available) or TOPKAT[32] and HarzardExpert[33] (commercial). The range of applied methods varies a lot. Toxtree is based on decision tree approaches, whereas Lazar combines statistical algorithms with regression [101]. TOPKAT uses statistical analysis of substructures which are associated with toxicity and applies QSAR equations [5]. HazardExpert makes use of expert knowledge-derived mathematical rules and artificial neural network predictions [5]. There are also tools specialised on ecotoxicity, focussing environmental effects, e.g. the Estimation Program Interface (EPI) Suite[34] and the Organisation for Economic Co-operation and Development (OECD) QSAR Application Toolbox[35] [101]. In connection to this issue, the Registration, Evaluation, Authorisation, and Restriction of Chemicals (REACH) legislation provides the freely available Reach-Serv tool for the prediction of chemical toxicology and environmental fate[36]. The data set analysed in this section only contains information about the lethal dose value LD_{50} (Section 2.3.3). Therefore, it is difficult to compare the approaches in this thesis with toxicological endpoint-related software solutions.

The focus of this project is the evaluation of the machine learning approaches decision tree, random forest, artificial neural network, and support vector machine in combination with different molecular descriptor sets. The main motivation for these investigations is based on the first results in my Bachelor Thesis [20]. A small

[30] http://toxtree.sourceforge.net
[31] http://lazar.in-silico.de
[32] http://accelrys.com/products/collaborative-science/biovia-discovery-studio/qsar-admet-and-predictive-toxicology.html
[33] http://compudrug.com/hazardexpertpro
[34] http://www.epa.gov/oppt/exposure/pubs/episuite.htm
[35] http://www.oecd.org/env/existingchemicals/qsar
[36] www.reach-serv.com/index.php

descriptor set was identified, leading to particularly good prediction results on part of an in-house database referring to intravenously applied substances. This outcome was confirmed, compared, and refined in L. M. Grüger's Bachelor Thesis [51] under my supervision.

3.3.2. Data Sets

For my Bachelor Thesis in 2009, a data set of around 1,000 very toxic and 1,000 non-toxic substances was generated with 324 molecular descriptors from OpenBabel [20]. The descriptor generation was done by B. Grüning[37] and is based on SMILES patterns of molecules referenced in the ChemIDplus search system (Section 2.3.3). This includes many different SMARTS patterns and several physico-chemical properties (Section 2.3.2). This two-class system was extended to a three-class system as shown in Table 3.14. The definition of toxicity classes was performed with the aim of simplifying the prediction model and generating equally sized classes.

Table 3.13.: Toxicity classes. LD_{50} values were splitted to three classes of around 850 molecules.

LD_{50} in mg/kg bw	toxicity class
$0.0 \leq x < 5.5$	very toxic
$300 \leq x < 500$	toxic
$850 \leq x$	non-toxic

Each toxicity class contained around around 850 molecules. Furthermore, a new descriptor set of 51 features was generated with QikProp (Section 2.3.2). For several structures, it was not possible to export descriptor values. Therefore, the QikProp data set contained less molecules (Table 3.14) at the end.

Table 3.14.: Toxicity class sizes. Due to unsolved conversion problems, the number of substances in the data set with QikProp descriptors was smaller than the one in the data set with 324 descriptors.

Toxicity class	324 descriptors data set	QikProp data set
Very toxic	850	458
Toxic	888	653
Non-toxic	875	320

[37] `http://www.bioinf.uni-freiburg.de/team.html`

3.3.3. Results

3.3.3.1. Workflow

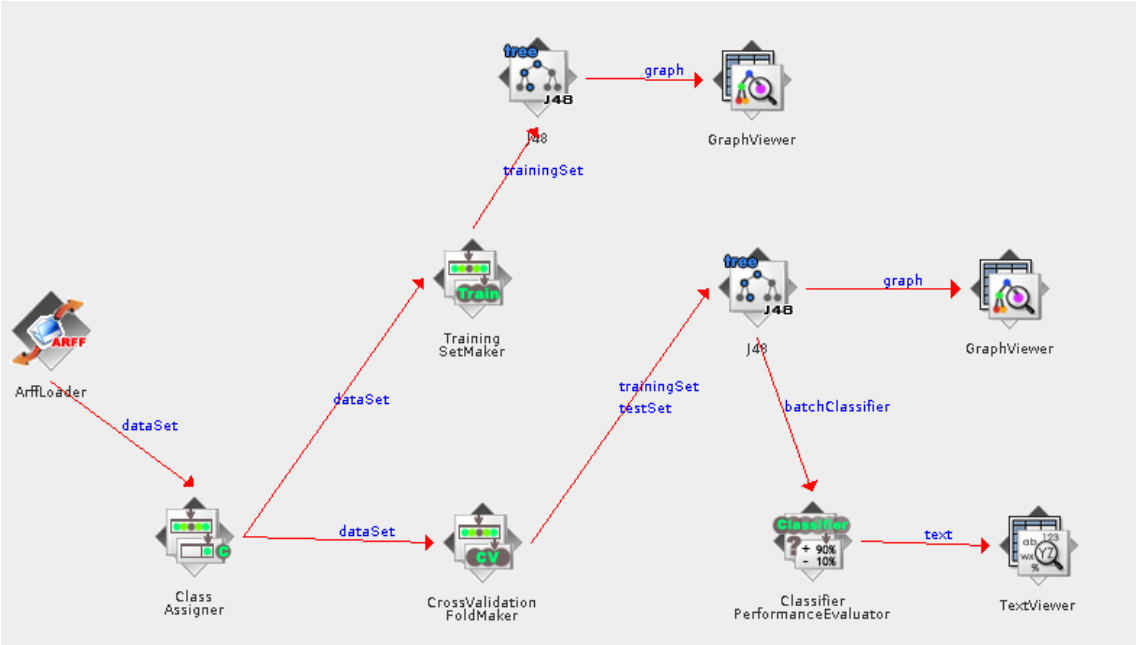


Figure 3.28.: WEKA KnowledgeFlow. The workflow starts with loading an input data set in the WEKA ARFF format with the ArffLoader. The classes to be predicted, e.g. very toxic and non-toxic, are selected with the Class Assigner element. The decision tree J4.8 can be built on the complete data set with the TrainingSetMaker and visualised with the GraphViewer element. The 10-fold cross-validation splits are performed with the CrossValidatonFoldMaker. The trainingSet and testSet data is classified with the decision tree classifier J4.8. Each single cross-validation tree can be visualised with the GraphViewer element. The results are summarised with the ClassifierPerformanceEvaluater. If a user wants to change the classifier, the decision tree J4.8 element needs to be replaced by another classifier element.

Every machine learning approach was applied to each data set with the WEKA KnowledgeFlow environment [25]. Figure 3.28 shows one of these workflows for the decision tree classifier. The CSV data sets were saved as WEKA-specific ARFF files with the Explorer environment (Section 2.2.8) and loaded into the machine learning workflow. The results are summarised in a text file and accessible via the TextViewer element. A Part of this output is shown in Figure 3.29. The most important evaluation parameters compared in this section are sensitivity, specificity, precision, recall, and AUC (Section 2.2.1). The specificity can be expressed as 100 %

minus false positive rate (FP rate). With this knowledge, the user can easily extract the results from the text file, also in the case of a three-class model. For all three-class models, the presented evaluation parameters correspond to the toxicity class very toxic. The other two classes are represented as a summarised class. A detailed calculation how to transfer a three-class confusion matrix to a two-class confusion matrix is explained in Section 2.2.1.6.

```
=== Evaluation result ===

Scheme: J48
Options: -C 0.25 -M 2
Relation: tox_10.csv.arff


Correctly Classified Instances        1634               62.5335 %
Incorrectly Classified Instances       979               37.4665 %
Kappa statistic                          0.4375
Mean absolute error                      0.3371
Root mean squared error                  0.4129
Relative absolute error                 75.8696 %
Root relative squared error             87.5973 %
Total Number of Instances             2613

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.627     0.15       0.669     0.627     0.647      0.773    nontoxic
                 0.658     0.123      0.729     0.658     0.692      0.834    very toxic
                 0.591     0.29       0.512     0.591     0.549      0.673    toxic
Weighted Avg.    0.625     0.189      0.636     0.625     0.629      0.759

=== Confusion Matrix ===

   a   b   c   <-- classified as
 533  80 237 |   a = nontoxic
  35 576 264 |   b = very toxic
 229 134 525 |   c = toxic
```

Figure 3.29.: WEKA text result. This is the result file of the BestFirst decision tree on the 10 descriptors data set. The accuracy is 62.5 %. Considering the very toxic class as the reference class, the sensitivity (recall) has got a value of 65.8 % and the precision a value of 72.9 %. The specificity can be calculated from the FP rate and equals 87.7 %. These values and the corresponding graphical tree can be found in Table 3.15 in Section 3.3.3.3. They are based on the confusion matrix, which shows the predictions as columns and the actual class (gold standard annotation) in the rows. The three classes are encoded as a (nontoxic), b (very toxic), and c (toxic).

3.3.3.2. Selected Descriptors

The 10 descriptors from my Bachelor Thesis in 2009 [20] are called Carboxylic_acid, Annelated_rings, Halogen_on_hetero, Hetero_N_basic_no_H, Sulfonic_acid, Primary_alcohol, Salt, Alkene, Tertiary_carbon, and CH-acidic_strong. Short explanations for all 10 descriptor names can be found in the Appendix (Section A.2). They were identified by a forward selection (Section 2.2.7.1) with the artificial neural network classifier on the data set with 324 descriptors, consisting only of very toxic and non-toxic instances. No cross-validation was performed.

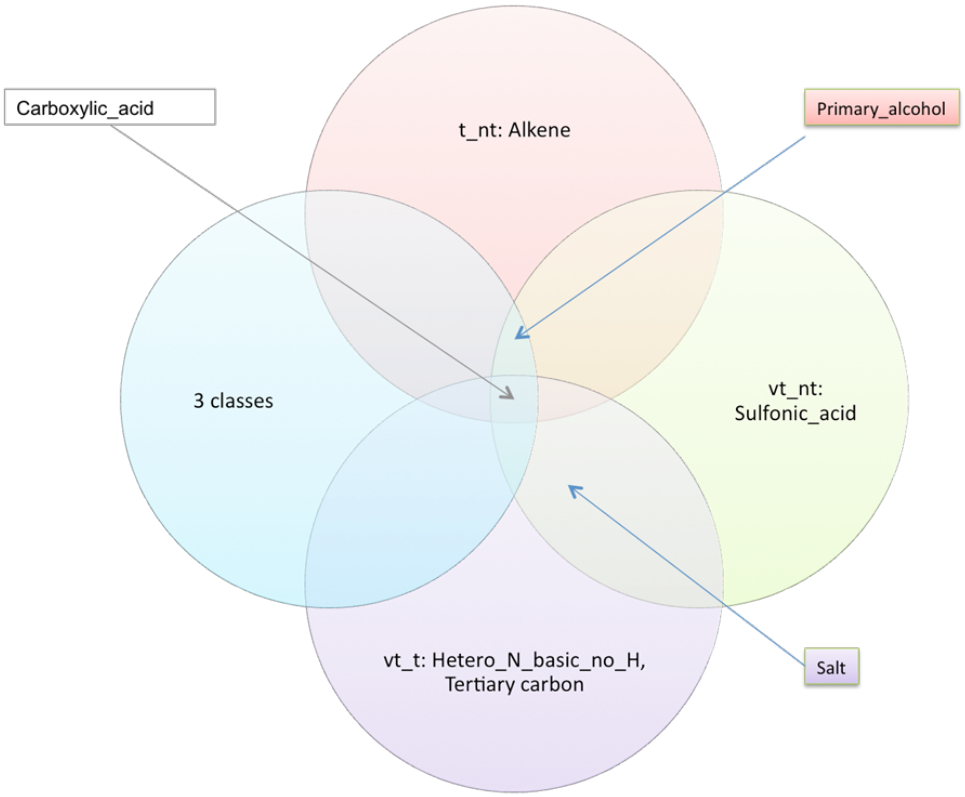


Figure 3.30.: Overlap of all BestFirst selections from 10 descriptors [51]. The figure is divided into four models, namely three two-class models and one three-class model. The two-class models are abbreviated with t_nt for toxic/non-toxic, vt_nt for very toxic/non-toxic, and vt_t for very toxic/toxic. For each circle, the descriptors contained in one or more class-models are shown. The descriptor carboxylic acid is contained in all models.

Given the subsets of instances for each toxicity class, a BestFirst search was performed on each two-class model and the three-class model to select descriptors that have a low feature-feature correlation and a high feature-class correlation

(Section 2.2.7.2). Figure 3.30 shows descriptors contained in one or more classes. From the 10 descriptors, the three features Annelated_rings, Halogen_on_hetero, and CH-acidic_strong, were not selected by BestFirst. The overlaps of the BestFirst selection on 324 descriptors and the QikProp descriptors are shown in the Appendix (Section A.1). Considering the BestFirst selection from 324 OpenBabel descriptors, Halogen_on_hetero and CH-acidic_strong were neither selected by any model. Every two-class model and the three-class model were tested on the data sets with 10, 324, and all QikProp descriptors, as well as their BestFirst subsets. Each of the presented prediction results was evaluated with 10-fold cross-validation. The two-class models are abbreviated as described in Figure 3.30.

3.3.3.3. Data Sets with 10 Descriptors

Table 3.15.: Decision tree J4.8 results on data sets with 10 descriptors. The model with the best AUC value is highlighed.

10D	J4.8	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	69.7	62.7	76.5	68.2	70.4
	BestFirst	67.7	61,1	74.1	66.5	68.5
Vt/nt	All	**84.0**	**85.2**	**82.9**	**85.2**	**88.3**
	BestFirst	81,7	87.3	76.3	86.1	86.8
Vt/t	All	76.8	83.0	70.5	80.3	79.7
	BestFirst	74.5	82.3	66.7	78.7	77.5
All	All	63.7	88.8	67.9	67.9	84.3
	BestFirst	62.5	87.7	65.8	72.9	83.4

The decision tree classifier accuracy ranges from around 70 % to 84 % (Table 3.15). The results with all descriptors are generally 2-3 % higher. The very toxic/non-toxic class shows the best results and the toxic/non-toxic class has got the worst performance. The three-class BestFirst model reflects the results in Figure 3.29. The tree can be seen in Figure 3.31. The decision tree model based on 10 descriptors is shown in the Appendix (Section A.4). In comparison, the BestFirst model uses five descriptors less (CH-acidic_strong not selected by the tree model based on 10 descriptors), but its performance is almost the same (Table 3.15).

On this descriptor set, the random forest approach shows a performance similar to the decision tree classifier, but with around 1-2 % better AUC values (Table 3.16).

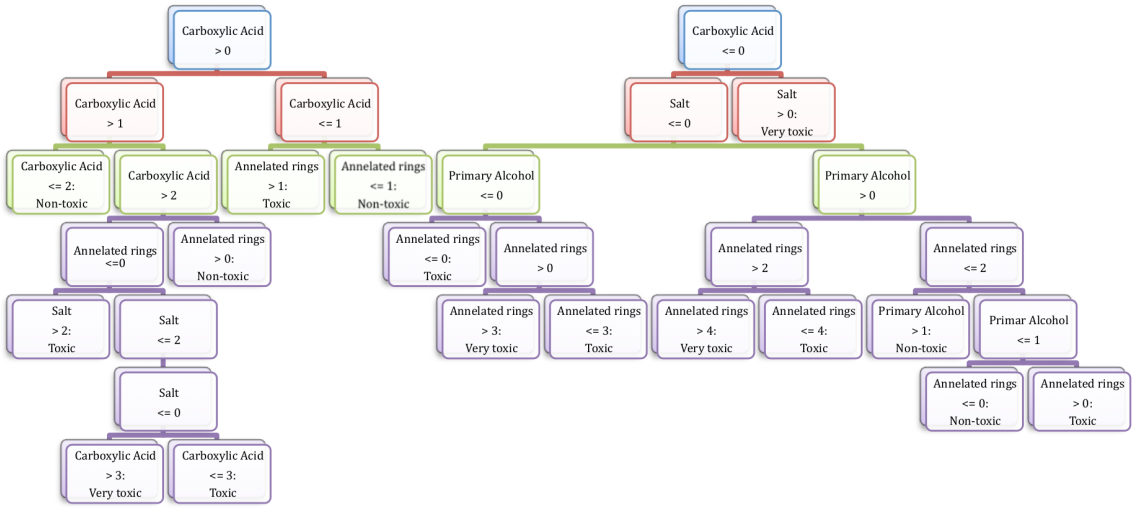


Figure 3.31.: Decision tree J4.8 with BestFirst selected features from 10 descriptors [51]. This tree belongs to the prediction result in Figure 3.29 and Table 3.15 for the BestFirst 10D data set. The descriptor carboxylic acid shows the root of the tree and is used several times to deal with an exact number of carboxyl groups (the left part of this figure).

Table 3.16.: Random forest results on data sets with 10 descriptors. The model with the best AUC value is highlighed.

10D	RF	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	69.4	62.0	76.5	67.8	72.9
	BestFirst	68.5	62.5	74.3	67.4	70.9
Vt/nt	All	**84.7**	**86.4**	**83.1**	**86.2**	**90.1**
	BestFirst	82.3	87.8	76.9	86.6	89.5
Vt/t	All	77.0	82.7	71.2	80.2	81.6
	BestFirst	75.0	82.5	67.4	79.2	79.1
All	All	64.1	88.1	70.1	74.8	85.5
	BestFirst	62.8	87.3	66.3	72.5	84.2

Artificial neural networks and support vector machiens also show a better performance with 10 descriptors. Their results are similar to the tree classifiers. In case of the support vector machine classifier, a diverse effect is shown. The specificity drops with less descriptors and the sensitivity increases. Although, the SVM classifier reaches the best accuracy value with 85.0 %, its AUC value is worst in this group. Therefore, the artificial neural network and random forest classifiers show be best performance. In general, the very toxic/non-toxic class shows the highest AUC values.

Table 3.17.: Artificial neural network results on data sets with 10 descriptors. The model with the best AUC value is highlighed.

10D	ANN	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	69.7	63.9	75.3	68.5	73.3
	BestFirst	69.0	63.3	74.5	68.0	70.5
Vt/nt	All	**84.3**	**85.3**	**83.3**	**85.4**	**90.0**
	BestFirst	83.0	82.7	83.2	83.2	89.3
Vt/t	All	76.1	83.6	68.4	80.4	81.9
	BestFirst	75.0	85.6	64.3	81.5	79.4
All	All	64.5	87.9	71.3	74.7	86.3
	BestFirst	63.6	87.3	68.3	73.1	84.4

Table 3.18.: Support vector machine results on data sets with 10 descriptors. The model with the best AUC value is highlighed.

10D	SVM	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	69.2	62.9	75.2	68.0	69.1
	BestFirst	68.1	62.4	73.6	67.1	68.0
Vt/nt	All	**85.0**	**86.2**	**83.8**	**86.2**	**85.0**
	BestFirst	80.6	74.1	86.9	77.6	80.5
Vt/t	All	77.3	86.5	67.9	83.2	77.2
	BestFirst	75.2	82.3	68.0	79.1	75.2
All	All	64.2	89.6	67.5	76.6	78.6
	BestFirst	63.2	87.6	66.6	73.0	77.1

3.3.3.4. Data Sets with 324 Descriptors

The decision tree model with 324 descriptors shows a better performance than the model with 10 descriptors, especially on the complete descriptor set (no BestFirst selection). The two-class model accuracies range from 72.5 % to around 90 %. The three-class model on all descriptors reaches an 8.8 % higher accuracy (Table 3.19).

The random forest classifier shows the best performance of all models with 324 descriptors. The AUC value of the very toxic/non-toxic model is 97.1 %. The accuracy values range from around 80 % for the toxic/non-toxic model to 91.3 % for the very toxic/non-toxic class and the three-class model reaches an AUC value of 94.4 % (Table 3.20).

The accuracy of the artificial neural network and support vector machine models is slightly worse than the one of the tree classifiers. Nevertheless, the BestFirst artificial neural network is superior to the decision tree in the AUC value as in the case of

Table 3.19.: Decision tree J4.8 results on data sets with 324 descriptors. The model with the best AUC value is highlighted.

324D	J4.8	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	75.1	74.7	75.5	75.7	78.8
	BestFirst	74.0	74.7	73.3	75.2	80.0
Vt/nt	All	90.1	90.2	89.9	90.5	90.1
	BestFirst	**86.8**	**91.5**	**82.3**	**90.9**	**91.8**
Vt/t	All	81.9	82.9	80.9	82.3	83.5
	BestFirst	76.9	92.9	60.7	89.3	77.9
All	All	72.5	89.0	81.4	78.8	86.9
	BestFirst	65.8	91.8	68.3	80.8	86.8

Table 3.20.: Random forest results on data sets with 324 descriptors. The model with the best AUC value is highlighed.

324D	RF	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	80.3	81.3	79.4	81.6	87.7
	BestFirst	74.6	74.8	74.4	75.5	82.0
Vt/nt	All	**91.3**	**93.4**	**89.3**	**93.3**	**97.1**
	BestFirst	87.2	89.6	84.8	89.4	93.8
Vt/t	All	85.6	85.5	85.8	85.3	93.3
	BestFirst	79.4	91.6	67.0	88.7	85.2
All	All	77.0	92.2	82.5	84.1	94.4
	BestFirst	65.6	91.1	68.1	79.3	87.7

Table 3.21.: Artificial neural network results on data sets with 324 descriptors. The model with the best AUC value is highlighed.

324D	ANN	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	49.7	99.9	0.16	93.3	65.6
	BestFirst	73.4	72.2	74.5	73.7	80.8
Vt/nt	All	82.0	74.8	89.0	78.4	88.8
	BestFirst	**87.5**	**87.8**	**87.3**	**88.0**	**94.0**
Vt/t	All	58.4	34.5	82.7	55.4	61.6
	BestFirst	79.5	93.5	65.3	90.8	85.5
All	All	51.0	58.6	75.7	47.9	77.0
	BestFirst	66.4	93.1	68.6	83.3	88.4

10 descriptors (Section 3.3.3.3). Compared to artificial neural network and support vector machine, the tree classifiers show the best performance with all available descriptors.

Table 3.22.: Support vector machine results on data sets with 324 descriptors. The model with the best AUC value is highlighed.

324D	SVM	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	70.3	76.0	64.9	73.8	70.4
	BestFirst	70.4	62.7	77.7	68.5	70.2
Vt/nt	All	81.7	86.1	77.5	85.2	81.8
	BestFirst	**86.7**	**88.2**	**85.1**	**88.2**	**86.7**
Vt/t	All	74.5	75.9	73.1	74.9	74.5
	BestFirst	76.6	94.0	58.9	90.7	76.4
All	All	63.0	88.1	64.7	73.3	76.4
	BestFirst	65.2	95.6	63.8	88.0	79.7

3.3.3.5. QikProp Data Set

The decision tree classifier shows lower AUC values with 51 QikProp descriptors than with 324 descriptors, although the difference within the very toxic/non-toxic model is only around 3 %. The sensitivity and precision values in the three-class model are considerably lower than the specificity values.

Table 3.23.: Decision tree J4.8 results on data sets with QikProp descriptors. The model with the best AUC value is highlighed.

QP	J4.8	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	74.3	70.1	77.3	78.6	71.4
	BestFirst	75.3	70.3	78.8	79.1	76.5
Vt/nt	All	87.8	89.5	85.3	85.0	88.4
	BestFirst	**83.9**	**81.2**	**87.8**	**76.6**	**89.8**
Vt/t	All	81.3	86.0	71.6	71.6	78.4
	BestFirst	79.3	90.3	56.9	74.3	78.5
All	All	67.7	91.4	65.3	68.5	79.8
	BestFirst	64.9	92.1	59.1	68.2	79.6

The performance of the random forest approach is the best in the group of QikProp data set models. The AUC values are 1-3 % lower than in the model with 324 descriptors, but clearly higher than with 10 descriptors. In this case, not only the sensitivity in the three-class model, but also in the very toxic/toxic model drops in comparison to a specificity of around 93-95 %. Nevertheless, the AUC values are in high range.

The results of the artificial neural network approach show a similar effect as the decision tree classifier, but its AUC values are clearly higher. The performance of the

Table 3.24.: Random forest results on data sets with QikProp descriptors. The model with the best AUC value is highlighed.

QP	RF	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	77.4	74.0	79.8	81.4	83.9
	BestFirst	76.1	72.5	78.7	80.3	84.2
Vt/nt	All	**89.7**	**92.1**	**86.3**	**88.5**	**95.2**
	BestFirst	85.6	89.1	80.6	83.8	91.8
Vt/t	All	84.2	95.6	60.9	87.1	89.4
	BestFirst	81.4	93.1	57.5	80.3	84.8
All	All	73.1	95.9	65.9	82.4	91.2
	BestFirst	70.1	95.7	59.1	79.7	89.6

Table 3.25.: Artificial neural network results on data sets with QikProp descriptors. The model with the best AUC value is highlighed.

QP	ANN	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	75.3	69.7	79.2	78.8	80.5
	BestFirst	76.9	68.1	83.1	78.8	81.3
Vt/nt	All	**88.9**	**91.5**	**85.3**	**87.5**	**94.0**
	BestFirst	83.5	81.6	86.1	76.5	90.7
Vt/t	All	83.1	90.2	68.8	77.5	86.1
	BestFirst	80.1	94.3	51.3	81.6	81.6
All	All	69.3	92.8	65.3	72.3	86.6
	BestFirst	66.6	95.4	45.3	74.0	84.6

toxic/non-toxic BestFirst Model is only 1 % worse than the random forest approach. In comparison of AUC values between the QikProp and 324 descriptor models, the artificial neural network shows similar results.

Table 3.26.: Support vector machine results on data sets with QikProp descriptors. The model with the best AUC value is highlighed.

QP	SVM	Accuracy	Specificity	Sensitivity	Precision	AUC
T/nt	All	60.1	4.4	99.2	59.6	51.8
	BestFirst	61.7	8.3	99.2	60.6	53.8
Vt/nt	All	59.9	99.8	2.8	90.0	51.3
	BestFirst	**81.7**	**88.2**	**72.5**	**81.1**	**80.4**
Vt/t	All	67.7	100.0	1.9	100.0	50.9
	BestFirst	67.9	97.1	8.4	58.7	52.8
All	All	46.9	99.9	2.8	90.0	51.4
	BestFirst	49.6	99.3	6.3	71.4	52.8

With the exception of the BestFirst very toxic/non-toxic model, the SVM did not learn an appropriate model of the classes showing specificity and sensitivity values below 10 % and an AUC value of around 50 %. Models with an AUC value in this range are considered as random classifiers (Section 2.2.1.5).

3.3.3.6. Summary

Random forests showed the overall best performance in the models with all 324 descriptors. Comparing all AUC values, the artificial neural network performed second best. In some cases, the artificial neural network accuracy values were inferior to decision trees, but this value can be optimised within the AUC computation (Section 2.2.1.5). In general, the QikProp very toxic/toxic models showed a tendency towards a lower sensitivity with a higher specificity in comparison to other descriptor sets. All classifiers performed best on the data sets with 324 descriptors and the QikProp models were superior to the models with 10 descriptors. The very toxic/non-toxic data set resulted in the best AUC values in each case. The tree models showed better results with more descriptors on the data set with 324 descriptors, while artificial neural networks and support vector machines showed a better performance with the lower number of descriptors in the BestFirst subset. The 10 descriptors were the result of a forward selection on the very toxic/non-toxic model. Nevertheless, the cross-validated model with the BestFirst selection from 324 descriptors showed better results than the cross-validated model with 10 descriptors. The support vector machine model showed good results as well, but it turned out to be the worst classifier in an overall comparison.

3.3.3.7. Principal Component Analysis

Every two-class and three-class model was exported with the first two principal components in WEKA and plotted in Python with Matplotlib. The data was not normalised. There was no graphical result showing a relatively complete separation of the scatter plots. Therefore, only the QikProp very toxic/non-toxic BestFirst model is displayed. In comparison, it showed the best separation (Figure 3.32).

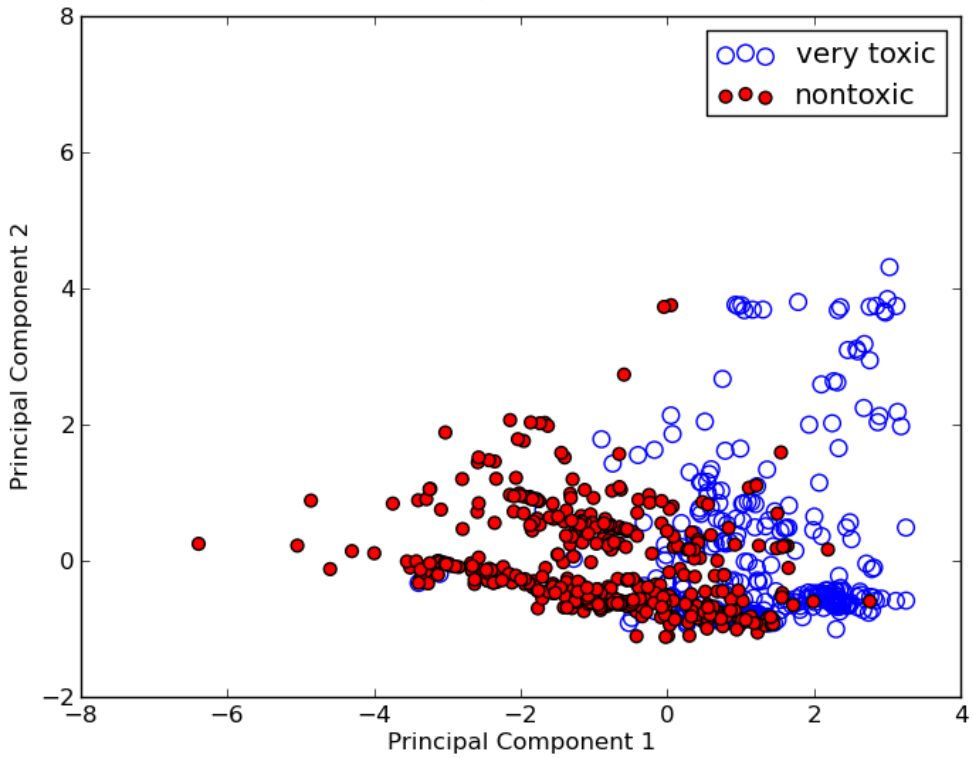


Figure 3.32.: Principal component analysis [51]. The first two principal components based on the QikProp very toxic/non-toxic BestFirst model are shown.

3.4. StreptomeDB

This section is related to a recently submitted publication in NAR Database Issues [42].

3.4.1. Related work

StreptomeDB was first published in 2013 [29]. This version contained more than 2,400 unique and diverse compounds from more than 1,900 different *Streptomyces* strains and sub strains. It turned out to be the largest database of natural products isolated from streptomycetes [29]. Furthermore, it contains information about compound activities and synthesis routes. The structures are freely available as downloadable SDF files. The in-house software Compound Research System Curator (CoRSCurator) was used to annotate these entities from thousands of automatically collected abstracts. The compound names were mapped to structures from PubChem [3] or drawn manually if not available. While the focus in the first version of StreptomeDB was on annotation of abstracts, the curators working on the new version always read the full texts of the article, if available. This approach results in a much higher proportion of read abstracts and extracted compound names with a source organism. Furthermore, there are new features such as an interactive phylogenetic tree for all available source organisms with sequences and information about annotated gene clusters [42]. The new Scaffolds Browser represents an advancement to the recently used most common substructure search (MCSS). It enables the user to explore the chemical space by displaying compounds with selected scaffolds and by composing them to higher-level ring systems [42]. The focus of my work in this new publication was on the creation, manipulation, and synchronisation of tables in the database back end. Therefore, this section focusses on the implementation of an update pipeline to include new information in StreptomeDB.

3.4.2. StreptomeDB Back End

The CoRSCurator is a standalone software written in Java, which provides a graphical user interface (Figure 3.33) to annotate abstract titles and texts with entities like compound name, source organism, activity, and synthesis route. This software

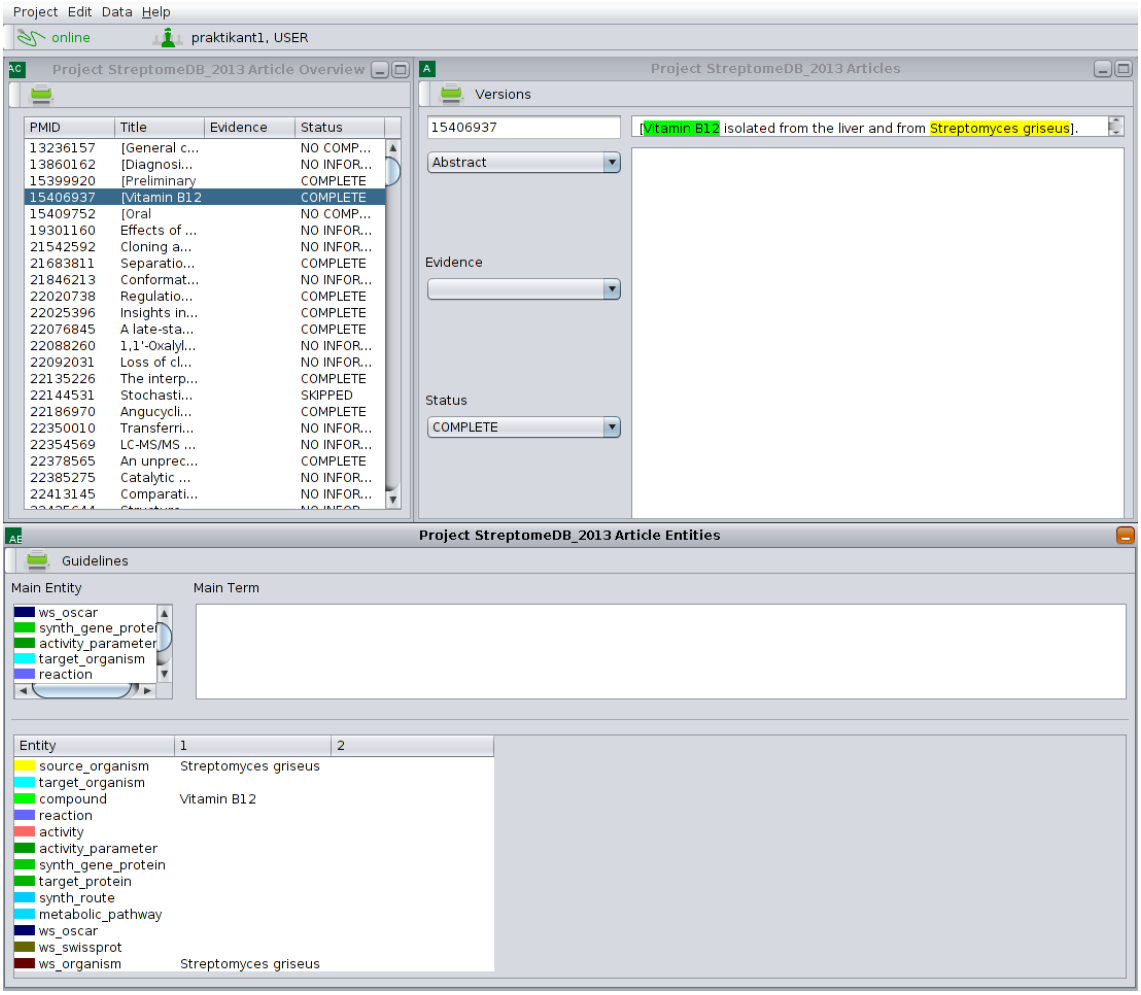


Figure 3.33.: CoRSCurator user interface. The user can select a PubMed article in the left upper window to display it in the right upper window. In this case, there is no abstract given for the PubMed ID 15406937. Nevertheless, the article title already contains a compound (vitamin B12) highlighted in green and the source organism (*Streptomyces griseus*) in yellow. In the bottom window, the user can enter annotations manually or by right-clicking the words directly in the given text. More windows can be displayed, layered, or resized as desired by the user. Every organism name will be coloured in yellow after annotating it as the entity source organism. The bottom window shows other types of entities which were not used by the curators in the recent version. It is possible to change the user praktikant1 to another user and to directly synchronise the appropriate user repository by fetching information from the PostgreSQL database corscurator_2013.

pushes its annotations to the PostgreSQL database corscurator_2013. All relations described in this database need to be processed, merged with non-redundant structural information, and uploaded to the PostgreSQL database streptomedb_2013. After synchronising the newly curated entites from corscurator_2013 with the data from the first StreptomeDB version, new PostgreSQL tables could be created to enable the storage of the features mentioned in the beginning. The official StreptomeDB homepage[38] provides a link to the new Web page[39], which shows the current front end of the features under development.

3.4.2.1. Compound Research System Curator Database

The CoRSCurator back end is based on 40 different tables. It was developed by C. Senger[40]. The few main tables which need to be processed by the update pipeline are explained here. Before a user can start to curate abstracts and to enter entity synonyms identified in the full text article, a data set needs to be uploaded to the corscurator_2013 database. This is done by the virtual user admin. A list of new PubMed IDs can be inserted into a special CoRSCurator window to be uploaded into the PostgreSQL database. This list can be fetched with PubMed2Go by selecting the non-intersecting PubMed IDs from the IDs that have already been uploaded. The search term for the list of PubMed IDs identified in NCBI[41] in the StreptomeDB project is *Streptomyces* in quotations.
The bottom window in Figure 3.33 shows a white box referred to as Main Term. In case of several organisms, each compound needs to be connected to its organism by right-clicking the selected terms. If there is more than one compound given and every compound has got an activity or a synthesis route, these entities also need to be connected manually. Therefore, two important tables exist in the corscurator_2013 database, named data_project_article_entity_term and data_project_article_entity_term_relationship. All entries that were not connected by the user will show up in the former table while the connected entities will be shown in both tables. A combination is also possible, e.g. one source organism and different molecules with different activities. In this case, the organism does not have to, but can be connected by the user. The following example tables illustrate this

[38] `http://www.pharmaceutical-bioinformatics.de/streptomedb`
[39] `http://132.230.56.145/streptomedb2`
[40] `http://www.pharmaceutical-bioinformatics.de/main/members`
[41] `http://www.ncbi.nlm.nih.gov/pubmed/?term=%22streptomyces%22`

issue in more detail. Table 3.27 shows a query to select PubMed IDs for which the compound chloramphenicol was annotated. The first PubMed ID from this query is taken to select the source organism as shown in Table 3.28. Table 3.29 and Table 3.31 show queries to find connected entities. Their results are shown in Table 3.30 and Table 3.32.

Table 3.27.: PostgreSQL query to select PubMed IDs from the table data_project_article_entity_term. This table contains all annotated entities that were not connected by the user. The query selects all references to the molecule chloramphenicol (part of the output shown).

SQL command	
SELECT	pmid
FROM	data_project_article_entity_term
WHERE	lower(term) = 'chloramphenicol';
- - OUTPUT:	23659856, 25267678, 23143535

Table 3.28.: PostgreSQL query to select terms from the table data_project_article_entity_term. This query shows all annotated entities with PubMed ID 23659856. The source organism *Streptomyces venezuelae* is shown, but there are no annotated activities and pathways.

SQL command	
SELECT	term
FROM	data_project_article_entity_term
WHERE	pmid = 23659856;
- - OUTPUT:	chlorampenicol, *Streptomyces venezuelae*

Table 3.29.: PostgreSQL query to select connected entities for PubMed ID 25267678. The article belonging to this PubMed ID contains the annotated entities activity and synthesis route for the molecule chloramphenicol as shown in Table 3.30.

SQL command	
SELECT	child_term, parent_term
FROM	data_project_article_entity_term_relationship
WHERE	pmid = 25267678;

Table 3.30.: Part of the result for the query in Table 3.29. The relationship between the child_term and the parent_term is not directed. In this case, the antibiotic activity of chloramphenicol and its synthesis route, the shikimate pathway, were annotated. Their explicit connection by the user can be considered as optional. The source organism *Streptomyces venezuelae* is specified with the strain description ATCC 10712.

child_term	parent_term
Chloramphenicol	Streptomyces venezuelae ATCC 10712
bacteriostatic	Chloramphenicol
shikimate pathway	Chloramphenicol

Table 3.31.: PostgreSQL query to select connected entities for PubMed ID 23143535. The article belonging to this PubMed ID contains many different source organisms as shown in Table 3.32.

SQL command	
SELECT	child_term, parent_term
FROM	data_project_article_entity_term_relationship
WHERE	pmid = 23143535;

Table 3.32.: Part of the result for the query in Table 3.31. This table shows that chloramphenicol is also produced by *Streptomyces coelicolor*. Furthermore, this is one of the source organisms of the compound actinorhodin. Several other source organisms and compounds are shown in this article.

child_term	parent_term
Actinomycin	*Streptomyces antibioticus*
Actinomycin	*Streptomyces parvulus*
Actinorhodin	*Streptomyces coelicolor*
Actinorhodin	*Streptomyces lividans*
Chloramphenicol	*Streptomyces coelicolor*
Erythromycin	*Streptomyces erythraea*
Streptomycin	*Streptomyces griseus*

3.4.2.2. Canonical SMILES

One major problem in the development of a synchronisation pipeline with old and new compound synonyms was their mapping to unique identifiers. This was achieved by using canonical SMILES. The curators had to work with a tabular file (Office sheet), containing the columns PubMed ID, molecule's name, PubChem ID (if available), and SMILES code. If a chemical synonym from an article is contained in

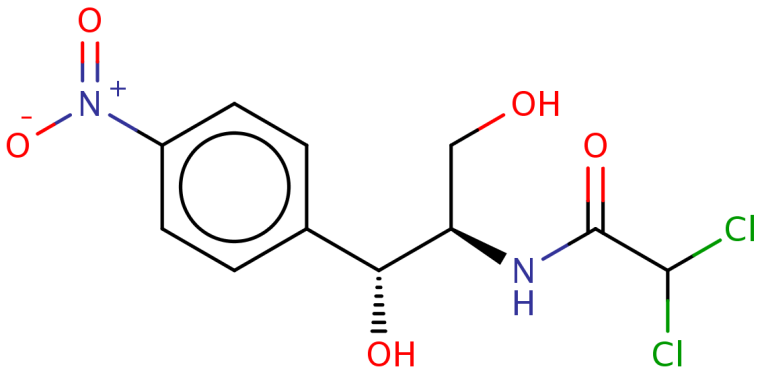


Figure 3.34.: Chloramphenicol. After downloading the PubChem SDF file with PubChem ID 5959, the picture was generated by pasting the SMILES code 'OC[C@H]([C@@H](c1ccc(cc1)[N+](=O)[O-])O)NC(=O)C(Cl)Cl" into MarvinSketch. In contrast to Figure 3.35, this SMILES pattern was generated using OpenBabel without the parameter "-b".

PubChem[42], the user just needs to add a PubChem ID and the SMILES pattern will be generated later. This is the case for chlorampenicol[43]. If this molecule would not be matched by name, the curator had to draw the structure manually from the figure in the paper and export its SMILES code, e.g. with MarvinSketch (Section 2.3.1), a software also used in this section. If there is no figure available in any publication, the molecule needs to be skipped. The compound chloramphenicol from Table 3.27 can be drawn as shown in Figure 3.34 or Figure 3.35. The appropriate SMILES patterns are shown in the figure subscriptions. Searching PubChem[44] with these SMILES codes will lead to the same molecule with the ID 5959. Using the downloadable SDF structure from PubChem, OpenBabel[45] [102] produces the canonical SMILES pattern "OC[C@H]([C@@H](c1ccc(cc1)N(=O)=O)O)NC(=O)C(Cl)Cl" with the command "babel -isdf 'infile'.sdf -ocan 'outfile'.smi -b". PubChem does not take care of stereochemistry. Therefore, Dr. Xavier Lucas[46] generated canonical SMILES codes for all drawn molecules by using a Galaxy pipeline[47] from the ChemicalToolBoX [103] and by visual inspection. The output of his mapping from synonyms to unique identifiers is a list of all curated PubMed IDs, chemical synonyms, PubChem IDs (if available), and canonical SMILES, exemplarily shown in Table 3.33.

[42] https://pubchem.ncbi.nlm.nih.gov
[43] https://pubchem.ncbi.nlm.nih.gov/compound/5959
[44] https://pubchem.ncbi.nlm.nih.gov/search/search.cgi#
[45] http://openbabel.org/wiki/Main_Page
[46] http://www.pharmaceutical-bioinformatics.de/main/members
[47] https://wiki.galaxyproject.org/Admin/GetGalaxy

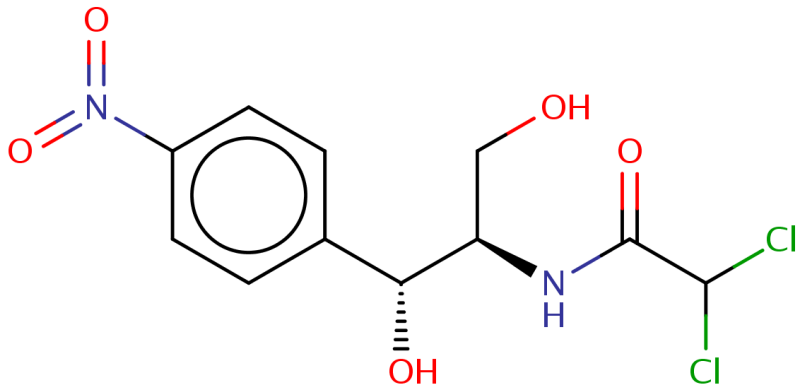


Figure 3.35.: Chloramphenicol with dative bonds. Referring to Figure 3.34, this picture was generated with the SMILES OC[C@H]([C@@H](c1ccc(cc1)N(=O)=O)O)NC(=O)C(Cl)Cl. The OpenBabel option "-b" for enabling dative bonds was used. Therefore, the nitro group differs in comparision to Figure 3.35.

Table 3.33.: Part of the molecules table with canonical SMILES. Synonyms only differ in a lowercase and uppercase letter. In general, database entries will contain several different synonyms for the same molecule.

PubMed ID	Synonym	PubChem ID	Canonical SMILES
23659856	chloramphenicol	5959	...(c1ccc(cc1)N(=O)=O)O)...
25267678	Chloramphenicol	5959	...(c1ccc(cc1)N(=O)=O)O)...
23143535	Chloramphenicol	5959	...(c1ccc(cc1)N(=O)=O)O)...

3.4.3. Data Integration to StreptomeDB Back End

The new structures summarised as shown in Table 3.33 needed to be integrated into the PostgreSQL database streptomedb_2013 and merged with the entities in corscurator_2013. The StreptomeDB back end works with structure files in mol format, stored in the PostgreSQL database via the plugin PGChem[48]. Each mol file contains a single structure saved in SDF format, without additional meta information. The development version was built with an older PGChem version[49] for PostgreSQL 8.4. This database can be stored as a PostgreSQL dump and reloaded to an empty database in PostgreSQL 9.4. SDF files can be generated from SMILES files with the OpenBabel command "babel 'infile'.smi 'outfile'.sdf". The PubChem parser developed for the web service CIL [1] also needs the meta information

[48] `https://github.com/ergo70/pgchem_tigress`
[49] `http://pgfoundry.org/projects/pgchem`

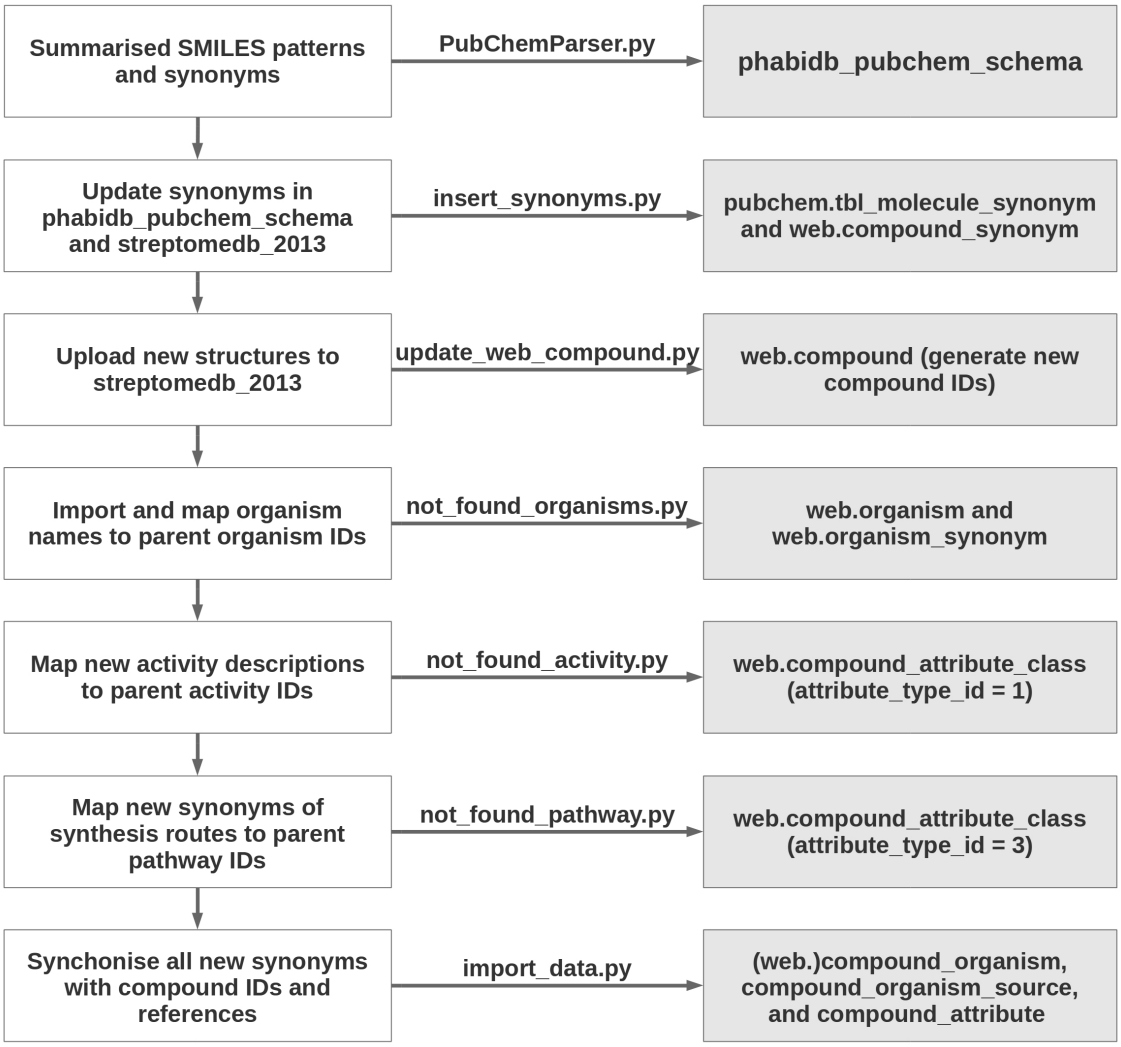


Figure 3.36.: StreptomeDB update workflow. From the summarised SMILES file, the database phabidb_pubchem_schema with the structures can be built and synchronised. Subsequently, the synonyms in stretomedb_2013 can be updated. After a semi-automatic update of organism synonyms, activitiy descriptions, and pathway names, these entities can be joined with the references in an iterative procedure until no more useful mappings can be performed. Activities and pathways are saved in the same table with a different attribute_type_id value.

"><PUBCHEM_COMPOUND_CID" which can be added with string concatenation. For structures without a PubChem ID, a negatively incrementing ID was generated (not shown on the Web page). A third database phabidb_pubchem_schema was built, containing all new structures and their PubChem synonyms[50]. This is the starting point for the automatic update steps as shown in Figure 3.36. The new

[50] `ftp://ftp.ncbi.nlm.nih.gov/pubchem/Compound/Extras/CID-Synonym-filtered.gz`

structures are uploaded to the streptomedb_2013 table web.compound with new compound IDs and all given compound synonyms are synchronised. All entity synonyms that could not be mapped automatically by the script import_data.py have to be mapped in an iterative way.
In case of new entity synonyms, the curator Dennis Klementz[51] was provided with a list of already existing parent IDs for each of the three entities organism, activity, and synthesis route. For the latter mentioned synonyms, a programme showed the new text elements one by one to type in the appropriate parent ID or an 'n' to skip the example. Considering the organism names, regular expressions are used to map part of a given synonym to existing strains in the database and to remove special characters that possibly hinder this process. If the basic strain is already known, the curator can commit the new strain and the parent ID of the base organism will be added automatically. If the synonym shows a completely new strain, it will be entered as such with the overall parent *Streptomyces*. An example is shown in Table 3.36, based on the queries in Table 3.34 and Table 3.35.

Table 3.34.: PostgreSQL query to select a compound ID. The compound ID of the molecule chloramphenicol can be queried by searching the compound table for the PubChem Compound ID 5959.

SQL command	
SELECT	compound_id
FROM	web.compound
WHERE	cid = 5959;
- - OUTPUT:	15

This update procedure resulted in the following new numbers of entities. The old number of 2,429 molecules increased by around 1,600 structures to 4,041 molecules. From the new data set, 853 molecules were drawn, which means they do not contain a PubChem ID. From 2,172 newly analysed articles, around 1,000 articles contained at least a mappable compound structure and a source organism. Furthermore, 1,732 new activity-compound relationships, 458 new pathway-compound relationships, and 599 new organisms were curated[52]. In the first version of StreptomeDB, around 10,000 abstracts were curated and 2,429 molecules could be found in around 3,700 abstracts.

[51] http://www.pharmaceutical-bioinformatics.de/main/members
[52] http://132.230.56.145/streptomedb2/statistics

Table 3.35.: PostgreSQL query to select organism names and IDs. The query in Table 3.34 returned the compound_id for the molecule chloramphenicol. The following query selects all organism IDs from the table web.compound_organism to find all organism names in the table web.organism and their IDs. The result is shown in Table 3.36.

SQL command	
SELECT	*
FROM	web.organism
WHERE	organism_id ;
IN	(
SELECT	organism_id
FROM	web.compound_organism
WHERE	compound_id = 15)
ORDER BY	organism_name ASC;

Table 3.36.: Results for the query in Table 3.35. Every basic strain has got the parent *Streptomyces* with the organism_id 15. It can be seen that the *Streptomyces venezuelae* sub strains have the parent ID 97. The parent of *Streptomyces venezuelae* ISP 5230-SVM1 is the strain *Streptomyces venezuelae* ISP 5230.

organism_id	organism_name	parent_organism_id
15	Streptomyces	
270	Streptomyces 3022a	15
62	Streptomyces coelicolor	15
1727	Streptomyces M-17633	15
137	Streptomyces phaeochromogenes	15
97	Streptomyces venezuelae	15
2084	Streptomyces venezuelae ATCC 10712	97
1372	Streptomyces venezuelae ATCC 15068	97
179	Streptomyces venezuelae ISP 5230	97
2639	Streptomyces venezuelae ISP 5230-SVM1	179

3.4.4. StreptomeDB Web Page

Information as provided in this section is summarised and displayed as shown in Figure 3.37, using the example of chloramphenicol[53]. The user receives details about the molecule itself, but is also able to compare these details with other molecules, e.g. by selecting other structures with the same scaffold. The new features of the StreptomeDB front end will be described in detail in the update publication [42].

[53] `http://132.230.56.145/streptomedb2/get_drugcard/15`

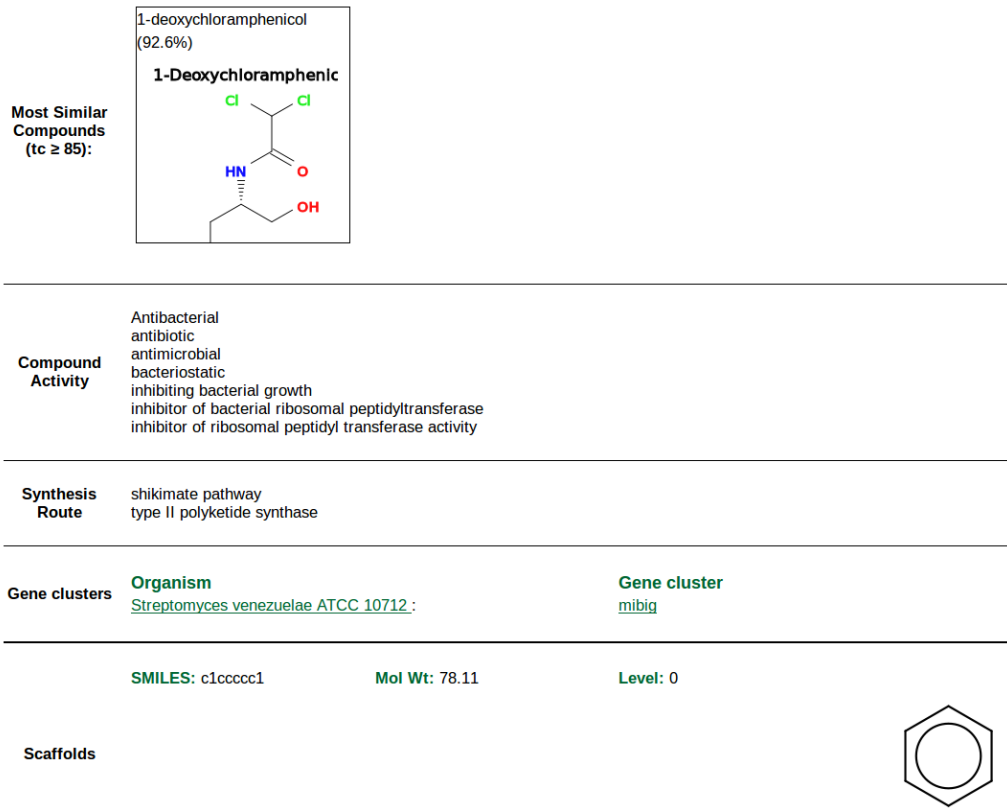


Figure 3.37.: Chloramphenicol in StreptomeDB. Several compound activities, two synthesis routes, and a link to the gene cluster are provided. The most similar molecule in the database is 1-deoxychloramphenicol with a Tanimoto coefficient of 92.6 %. Another part of the compound view not shown in this figure refers to molecular descriptors, such as number of atoms (#Atoms) and the topological polar surface area (TPSA). Furthermore, the canonical SMILES pattern is provided. This molecule only contains the benzene ring as a level-0 scaffold, which can be combined with other scaffolds by clicking the displayed structure.

4. Discussion and Future Prospects

4.1. Development and Usability of PubMed2Go

The selection of Python, PostgreSQL, and Xapian is based on the requirement to provide a framework that is easy to install, independent from a Web service, and usable with only a few lines of code. Using another type of database management system such as Oracle or MySQL with the object-relational mapping approach is possible (Section 2.1.2), as well as using the Xapian interfaces for Java or C++ instead of Python. PubMed2Go enables the user to directly create individual queries and to extend this framework with sophisticated approaches from natural language processing such as part-of-speech tagging [85] and named entity recognition with BioC [81, 82] or dependency parsing [9]. The PubMed2Go database can be used as a centralised repository, which is updated weekly. This also offers the possibility to monitor changes in a data set over time by executing a workflow repeatedly. Furthermore, PubMed2Go can be extended to use BioC XML PMC documents[1]. The combination of a relational database and a full text index can also be integrated in a web server, as shown in previous publications [1, 2, 29]. In these projects, the PubMed2Go framework was extended with named entity recognition for gene or protein names from UniProt, PubChem synonyms for small molecules, and organism names. The web servers are based on the combination of the object-relational mapping approach from PubMed2Go with a Django interface in Python. This emphasises as well that PubMed2Go can be used in a modular way depending on the user's needs. Therefore, PubMed2Go can be helpful in future BioCreative challenges[2], too.

The mapping of chemical structures and protein sequences to synonyms of biomolecules enables the possibility of linking knowledge from publications to specific models built

[1] `ftp://ftp.ncbi.nlm.nih.gov/pub/wilbur/BioC-PMC`
[2] `http://www.biocreative.org/events/biocreative-v/CFP`

on data sets of biomolecules and vice versa. Synonyms can be redundant. So far, curators are needed for high quality data sets as in the case of StreptomeDB. Models improved a lot, e.g. comparing Whatizit and GeneTUKit [43]. With PubMed2Go, one more step into the process of bridging the gap between a large amount of data in PubMed and plenty of available recognition and relation extraction approaches has been made. The software library offers a ready-to-start solution for developing large-scale text mining applications by generating an in-house database from PubMed articles. The resulting data environment supports complex relational database queries and fast full text search. The BioC interface and the possibility to use Docker provide interoperability to apply NLP approaches in different programming languages and to run queries on several operation systems without much programming effort. All software and included methods are open-source and free to be modified for further refinements and improvements within the community.

4.2. Model Integration of Functional Relationships in Texts to the Web

In the Tikk *et al.* benchmark of different kernel methods on different protein-protein interaction corpora, the F_1 score ranged from 54.5 % to 74.5 % in case of the shallow linguistic kernel and from 56.2 to 78.1 % in case of the all-paths graph kernel, without notable drops in specificity or recall [9]. Segura-Bedmar *et al.* showed recall values from 69.6 % to 78.6 % and precision values from 42.4 % to 52.1 % on a newly created drug-drug interaction corpus [12]. The resulting F_1 scores were in a range from 54.9 % to 59.6 % for the same shallow lingustic kernel parameter selections as in this thesis. The drug-drug interaction corpus was automatically built from a set of 579 documents. Subsequently, it was annotated with drug-drug interactions by a researcher with pharmaceutical background. Similar to the compound-protein interaction corpus in this thesis, Segura-Bedmar *et al.* did not determine an explicit type of interaction. In case of the protein-protein interaction corpora, part of the data sets is based on affinity values and directed interactions [54].

It is important to know the baseline of the co-occurrences in the corpus, which is defined as the number of positive examples diveded by all available instances. This is one main criterion to select or decline machine learning approaches, because the F_1 score can be calculated for the concept of co-occurrences with this number (precision)

and a recall of 100 % (Section 3.2.2.3). In case of the drug-drug interaction corpus, only 10.3 % of all 30,757 candidate interactions were annotated as positive examples. With this precision, an F_1 score of 18.6 % is calculated. While the co-occurrences approach on the compound-protein interaction corpus shows an F_1 score of 76.4 % on data set 1 (with interaction verbs) and 70.5 % on data set 2 (without an interaction verb enclosed), as shown in Table 3.3, the F_1 scores in the five protein-protein interaction corpora are 30.1 %, 41.7 %, 55.4 %, 57.6 %, and 70.3 % [9]. Although it is not reasonable to select an approach with a low precision and a recall of 100 %, that always classifies functional relations as positives by definition, it is important to compare these F_1 scores with the ones of the machine learning approaches.

The shallow linguistic models showed different results for the models choosing so called GENIA chunk tags and models considering all words in a sentence with punctuation. These chunks represent a possibility to connect terms to short phrases, e.g. "NF-kappaB activation" or "the anti-apoptotic effect" (Figure 3.19 in Section 3.2.5).

The parameters window size 3 and the n-gram value 3 for shallow linguistic kernel resulted in the same F_1 score as in the co-occurrences approach on data set 1 (0.3 % better model with chunking). Therefore, the shallow linguistic kernel approach reached a clearly better precision than the co-occurrences, and a good recall of 81.9 % using chunk tags and 78.6 % without. Therefore, less of the actual true interactions are recognised by the model with the lower recall, but its specificity and precision are a few percent higher than in the other model. The precision reflects how many of the positively predicted values are real interactions. The specificity describes how many of the negative examples were predicted as no interactions. Therefore, the model using all words of a sentence and punctuation with the lower sensitivity can be preferred over the model using chunks. Unfortunately, the AUC values cannot be provided in this version to make a clear model selection. The reason is that the standard implementation of LIBSVM does not return a probability score [64]. This can be solved easily by using the modified Tikk *et al.* implementation. Furthermore, the Tikk *et al.* pipeline makes use of document-level cross-validation, which was not encoded for the shallow linguistic kernel preprocessing within this thesis (Section 2.2.2).

Considering the shallow linguistic kernel results on data set 2 in detail, the decision whether to take the model with or without chunking is obvious, because the prediction including all words of a sentence and punctuation shows a better performance for

all evaluation parameters. This fits to the statement of Giuliano *et al.* that their results turned out to be worse in case of excluding stop words and punctuation [63]. Comparing this model to the co-occurrences approach, the precision is around 23 % higher and the F_1 score around 9 %. One main difference between the shallow linguistic kernel results on data set 1 and 2 is that the specificity on the data set without interaction verbs is around 15 % higher than in the data set containing interaction verbs.

Considering all evaluation parameters, the shallow best linguistic model on data set 2 without chunking performed up to around 2 % better than the all-paths graph model with the regularisation parameter value 2. In general, the high co-occurrence baseline and the comparably low specificity values on data set 1 for both kernels indicate that sentences with an interaction verb are more difficult to classify than sentences without this structure. On data set 1, the all-paths graph kernel reaches an around 6 % higher sensitivity and 2 % better precision value than the shallow linguistic kernel, resulting in a 4 % higher F_1 score. Therefore, the all-paths graph kernel approach is able to make use of the interaction verbs as a deep parser. One reason for the lower sensitivity of the shallow linguistic model on data set 1 could be that it is restricted to a word neighbourhood of up to three words around the target. Therefore, distant relationships in sentences with different and not directly dependent substructures are more difficult to predict for this kernel. Nevertheless, the all-paths graph kernel's specificity of 56.3 % is only 0.2 % higher than the one of the shallow linguistic kernel model without chunking. The higher number of positive examples in data set 1 also represents a possible reason for the relatively high sensitivity, but low specificity in both models.

Segura-Bedmar *et al.* stated that a higher value of n-gram led to a higher precision and a lower recall. In contrast, the window size parameter did not show a visible effect within the different results. In case of classifying functional relationships between compounds and proteins, the window size also does not seem to have a strong influence on the performance, although in general the best results are achieved with both parameters set to the value 3. Except for the data set 1 model without chunking, the precision even increases slightly. In the results part, it was already mentioned that the lower n-gram values lead to a higher specificity on the compound-protein interaction corpus, but this finding cannot be compared with the drug-drug interaction corpus, because this information is not provided in the article. One reason why changing the window size did not really show an influence on the classifier's

performance is that this parameter was especially used to provide information about the direction of a relationship [63].

The AUC values within both all-path graphs kernel models for all four regularisation parameter selections do not differ by more than 0.5 %. A higher regularisation parameter generally represents a lower risk of overfitting (Section 2.2.4.5), but in this case, no clear conclusion can be drawn for the selection of this value. The all-paths graph kernel implementation of Airola *et al.* optimises the threshold for the highest F_1 score. In general, this might result in a rather high threshold selection, which is reasonable for the classification of functional relationships in texts (Section 2.2.1.5). However, the F_1 score is also known to be sensitive for class skew [9, 54]. Therefore, the optimal threshold should be further evaluated by directly comparing the evaluation parameters of F_1 score and accuracy.

The Tikk *et al.* benchmark also showed, that the best performing kernel methods cannot be considered as significantly better than a simple rule-based approach as implemented in RelEx [104]. This is an important finding, because a rule-based approach does not need any extensive training and no parameter tuning as well. As future prospects, this approach can also be compared to the applied kernel methods on the compound-protein interaction data set.

To conclude, it is important to compare the runtime of the models and the proportion of sentences with and without an interaction verb structure in PubMed. The former topic was already mentioned in the results part with the outcome that the all-paths graph kernel is up to four times slower than the shallow linguistic approach. These runtime measurements refer to a 10-fold cross-validation model. The limiting step is not the prediction, but the preprocessing of the sentence structures. The question of how many sentences exist in PubMed that contain the verb structure defined here, cannot yet be answered. This number will be computed for the planned publication of the evaluated models. The assumption is that the number of sentences without such a structure is much higher, because the original size of data set 1 was 2,964, reduced to 1,259, and in case of data set 2, there were 5,365 sentences, reduced to 1,494. The reason for this rather large decrease in the amount of used sentences is that there were so many false positive protein examples from the Whatizit pipeline (Section 3.2.2.1 and Section 3.2.2.2). Therefore, the original number of sentences in data set 2 was twice of the amount in data set 1, but the final number of sentences turned out to be quite similar. It is not possible to estimate the number of false positive examples in

the complete PubMed data set, but it is reasonable to assume a more reliable number from the new prolific version. In the new database version, GeneTUKit will be used to identify gene and protein synonyms in an organism-specific way. The tool reached a performance among the best results in the BioCreative III gene normalization task [43]. A prototype implementation for the prolific update already exists, making use of the PubMed2Go infrastructure. This also means, that the applied compound named entity recognition approach, which is based on the Hettne-Rules [49], has to be evaluated, e.g. on the SCAI corpus[3]. Furthermore, the approach in prolific has to be compared to other chemical compound named entity recognition approaches, e.g. ChemSpot [105]. During the annotation process of data set 1 and data set 2, several words were identified which usually lead to a false positive hit, when following compound names. An example of such a blacklist word was already shown in Figure 3.15 in Section 3.2.3.3.

Considering all results, different scenarios are possible for an integration of the prediction models to the Web service prolific. It is possible to use the all-paths graph kernel for sentences as they occur in data set 1 and the shallow linguistic kernel for sentences as they are contained in data set 2. An alternative is to create a stratified data set from data set 1 and data set 2, sampling the ratio of sentences with and without interaction verb, as they appear in PubMed. This model can be further evaluated on a newly selected data set from PubMed of another 1,000 sentences, also consisting of the structure with and without interaction verb in a stratified way. The performance of the best classifier can be the criterion to apply its generalised model to the whole PubMed under consideration of the complete runtime including all preprocessing steps.

Cross-validation results with an F_1 score of around 80 % represent a good performance within the research area of text mining. Nevertheless, the new model to predict functional relationships between compounds and proteins should be considered as a filter option in the prolific Web service to significantly decrease the amount of sentences, a user has to read through after sending a specific query. The aim is to simplify and speed up the selection of correct functional interactions. Another question that can be answered with this approach is whether a user can directly search for rare compound-protein relationships that have a low frequency. This

[3] `http://www.scai.fraunhofer.de/en/business-research-areas/bioinformatics/research-development/information-extraction-semantic-text-analysis/named-entity-recognition/chem-corpora.html`

problem cannot be solved with the approach of co-occurrences. The issue can be refined by using the BioInfer relationship ontology from Pyysalo *et al.* [106]. In connection to this, it can be evaluated how many verbs of each type have been used in data set 1, data set 2, and the overall PubMed data set. In either case, the selection of the model to be implemented has to be made between the protein-protein interaction benchmark implementation, the shallow linguistic kernel pipeline implementation, and a new pipeline implementation from scratch, e.g. with the BioC interchange format. Within the project time, the work on the all-paths graph kernel pipeline started almost a year after the code refinements of the shallow linguistic kernel pipeline. This is the reason for the quite different implementation of the preprocessing steps. In general, the selected procedure of training a model with the shallow linguistic and all-paths graph kernel can also be tested on annotated data sets with other types of relationships, e.g. gene-disease relations.

Although the performance of machine learning models in text mining approaches is still worse than the ones in other research areas, such as toxicity prediction, the application of sophisticated algorithms like kernel methods can be considered as beneficial for users from the life sciences. The approach presented herein is restricted to single sentences. It is known, that language can build complex structures, but only the simple cases are easy to recognise. At the moment, prediction models are strongly dependent on accurate training data set annotation and test data curation, but there is a continuously growing support for large-scale text mining and natural language processing applications within the community. A rather new approach in this research direction is called distant supervision [107]. It is based on annotating only part of a huge data set, learning graph-based patterns, using an ontology, and applying the patterns to the rest of unstructured data with machine learning. Other approaches to support the process of text processing and analysis are clustering, especially using topic models [108], or crowdsourcing [109]. Possibilities are manifold. The task of enhancing natural language processing models and applying them to the vast amount of available data is a challenging, but feasible aim.

4.3. Towards Understanding Toxicity Prediction

A prediction accuracy of around 80-90 % is a very good result, but it still cannot be used to give advice for the consumption of a possibly very toxic substance, even in a

low dose. Therefore, *in silico* approaches will not replace animal tests, but they can be used to generate warnings in the process of drug development or the research area of ecology in an early stage to save animal lives. The AUC value of 97.1 % for the very toxic/non-toxic model of the random forest classifier is outstanding. One reason why the other evaluation parameters of this model are in a range of around 91 % to 93 % is probably the standard decision threshold of 0.5 to separate two classes. Given the distribution of probability scores with the AUC, this threshold can be optimised for a higher accuracy, as explained in Section 2.2.1.5. Another important step is to further evaluate the set of descriptors that was chosen within the random trees of this classifier and to compare it with the BestFirst selected descriptors. The random forest classifier also performed best on the other three models with 324 descriptors, reaching an AUC of 87.7 % on the toxic/non-toxic model, 93.3 % on the very toxic/toxic model, and 94.4 % on the three-class model.

Several technical alterations in the models will possibly lead to refined results. First of all, none of the machine learning algorithm was varied in the range of specific parameters. Furthermore, the selection of descriptors with the filter method BestFirst for the data sets with 324 descriptors led to better results in case of the artificial neural network and support vector machine classifiers, but to worse results in the case of decision trees and random forests. The 10 descriptors used in this thesis were selected by a forward selection (Section 2.2.7.1) with an artificial neural network without cross-validation [20]. This approach is referred to as a wrapper method, because the selection of descriptors depends on the performance of the classifier (Section 2.2.7). The procedure should be repeated with the settings in this thesis and combined with a backward elimination (Section 2.2.7.1). This procedure counteracts the problem of local minima. The BestFirst filter tries to find the best descriptor set for all classes (Section 2.2.7). Therefore, the classifiers' performance should be evaluated with the BestFirst feature set of the three-class data set, applied to all two-class models. Another question is how similar the compounds in a cross-validated training and test data set are. One approach to ensure diversity is to cross out every molecule from a pair of most similar chemical substances until a certain threshold. Machine learning algorithms should be trained with a minimum of instances that maximally represent the different features in the data set. An approach to randomise the selection of training instances is bootstrapping, directly used within the algorithm of random forests (Section 2.2.5.2). The problem of selecting descriptors with filter and wrapper methods will probably change by extending and

changing the different toxicity classes. More descriptors need to be generated, e.g. with RDKit[4] or the Chemistry Development Kit[5] (CDK). The features used in this thesis were not normalised, because the range of physico-chemical descriptor values did not show large divergences. Using a principal component analysis as shown in Figure 3.32 did not lead to a clear class separation in the scatter plot, but this might change with different descriptors. Normalisation will also have an effect on the selection of principal components. A rather different approach is to transform the classification problem into a regression problem [25]. The QikProp descriptors also led to promising results, but a further evaluation requires the fixing of technical problems in the feature generation process for several substances. An approach different to the selection of functional group and physico-chemical descriptors is the consideration of the topological information about a molecule. This includes structural features from 2D fingerprints [110] and a range of other descriptors, e.g. topological pharmacophore (CATS) descriptors[6] or Molecular Operating Environment (MOE) descriptors[7] [111, 112, 113]. Furthermore, self-organising maps can be used to learn a structurally active pattern and display it in 3D [114, 115]. With these descriptor sets, not only the information about the existence of a functional group is considered, but also its position and its neighbourhood within the molecule. These changes probably lead to the identification of new patterns from different descriptors to be used for classification. Therefore, extensive testing of new models is needed to elucidate the recent findings.

Although the presented results are promising, it has to be mentioned that the prediction model is strongly simplified. The outcome needs to be discussed in a follow-up project meeting with toxicology experts to find explanations for the findings. The BestFirst selection identified the carboxyl group as the overlapping descriptor in all models, considering the data sets with 324 descriptors and the QikProp descriptors. A rather old publication from 1892 states that if an oxygen satured carboxylic acid group is introduced to a toxic, aromatic substance, it will be relatively less toxic due to the ability of not being reducible in the organism [20, 51, 116]. Furthermore, the 10 selected descriptors from the forward selection [20], shown in Section 3.3.3.2, are polar functional groups. Xenobiotics are defined as foreign substances or molecules in an unusual high concentration in the organism. It is known, that lipophilic

[4] `https://github.com/rdkit`
[5] `https://github.com/cdk`
[6] `http://gecco.org.chemie.uni-frankfurt.de/cats_light/index.html`
[7] `https://www.chemcomp.com/MOE-Medicinal_Chemistry_Applications.htm`

xenobiotics tend to be less toxic after a Phase I reaction with the Cytochrome P450 enzyme system (CYPs) mainly provided by the liver. In a Phase I reaction, e.g. a hydroxyl (-OH) can be added to a molecule such that it becomes more water-soluble [117]. Therefore, the question arises whether the descriptors found in the very toxic/non-toxic data set are actually encoding for toxicity or non-toxicity. A possible explanation is that substances with polar functional groups can be classified easier as non-toxic because of renal excretion. There are effects described in ChemIDPlus for certain substances, but these toxicological endpoints have not yet been considered. Another approach is to directly use software predictions of how the molecule is metabolised by CYPs, e.g. with SMARTCyp [118]. This might lead to a clarification of toxic effects by experts. The TOXNET search system[8] can be useful in this process, too. TOXNET covers a range of literature and structure repositories related to toxicology, e.g. HSDB[9]. Finally, this question leads to the requirement of recurrent collaborations with chemists and pharmacists to interconnect the different working fields and data sources in life sciences. Therefore, it is especially reasonable to aim for the integration of the developed models to the OpenTox community[10].

4.4. A Compound-centralised View on Streptomycetes

The process of manual text annotation and assigning of structures has led to a considerable amount of newly structured information about *Streptomyces* strains and sub strains. Nevertheless, this process can be error-prone. The only possibility to keep a good quality in the database is to turn this procedure into a peer-review process. All computationally supporting steps are provided within this thesis.

The recognition of chemical structures in publications, e.g. with the Open-Source Chemistry Analysis Routines software (OSCAR) [119], is not yet on a level to be used in a fully automated way. Therefore, this approach was not used in StreptomeDB. Furthermore, the results of the update pipeline showed that the outcome of reading the paper is much higher than of the procedure used in the past, focussing on abstract texts and looking only partially for information in the full texts (if available).

[8] `http://toxnet.nlm.nih.gov`
[9] `http://toxnet.nlm.nih.gov/newtoxnet/hsdb.htm`
[10] `http://www.opentox.org`

Considering the first version of StreptomeDB, there are around 5,000 PubMed IDs, which have not yet been curated. Another 5,000 PubMed IDs were not selected in this first version, because their PubMed database entries did not contain an abstract. On average, every second full text contained at least a source organism and a chemical compound. In the database version from 2013, this ratio was slightly more than one third. Currently, it can be assumed that up to 1,000 new articles need to be curated per year, based on a search for "Streptomyces" in PubMed with the restriction of publications from 2014. Furthermore, it can happen that PubMed IDs change over time. In the local database, they are still available, but in PubMed these articles have to be searched via title, not via PubMed ID.

Every new update data set can be processed with the pipeline shown in Figure 3.36, if there is a summarised SMILES file containing all new synonyms and references. Despite the supporting usage of ChemicalToolBoX, the task of judging the uniqueness of a molecule's structure and its mapping to a canonical SMILES pattern will probably be realised by experts with a chemical or pharmaceutical background. Every curator can check whether a possibly new molecule is already in the database by searching the canonical SMILES pattern in a simple text file with all 4041 old and new molecules contained in the recent version of StreptomeDB. The results show, that many new structures can be found by carefully reading all full texts matched by the abstract search, but this process is also much more time-consuming than only considering abstracts.

The new features on the StreptomeDB Web page represent a completely new data repository about streptomycetes. Users with a chemical or pharmacological background can use the scaffold search system to group molecules based on different levels of ring systems. Biologically interested researchers can identify *Streptomyces* strains in relation to a phylogenetic tree of all so far curated organisms that could be mapped to a genome. This information can be connected to the available gene clusters linked in StreptomeDB. A detailed discussion will be found in the update publication of StreptomeDB [42]. The process of further developments will be community-driven, which means that the user should actively propose new ideas for features to be implemented. Furthermore, it will be interesting to see how scientists make use of this project. Hagan *et al.* took a subset of compounds from StreptomeDB and several other repositories to compare their similarity to known metabolites from the community reconstruction of the human metabolic network (Recon2 [120]) [121]. Lucas *et al.* collected more than 68 million unique molecules and studied there

physico-chemical properties, stating which compounds are natural products-like, fragment-like, inhibitors of protein-protein interactions like, and drug-like [122]. StreptomeDB metabolites were also part of this analysis. F. Ntie-Kang investigated the drug-likeness of all StreptomeDB compounds with Lipinski's rule of five [123] and predicted their absorption, distribution, metabolism, elimination, and toxicity (ADMET) with physico-chemical properties and molecular descriptors [124]. Further scientific findings will follow by studying these versatile organisms within different life sciences.

4.5. Conclusion

This thesis shows that the combination of methods from text mining, cheminformatics, and machine learning with curation and annotation leads to reasonable applications and findings. Nevertheless, this progress can still be considered to be in an early stage. PubMed2Go can be considered as the basic element in this thesis, because it generates the necessary infrastructure to connect the large amount of data available in PubMed to the research area of text mining, including machine learning approaches and cheminformatics methods. All ongoing parts in this thesis are related to small molecules. The part of the StreptomeDB project which is described here covers the annotation and synchronisation of curated data sets. The toxicity prediction project successfully investigates high-dimensional representations of chemical compounds with molecular descriptors, based on reliable data, which was also extracted from literature by scientists. These dose-related models can be further evaluated and applied to other data sets, e.g. the natural compounds annotated in *Streptomyces* strains. The prediction of functional relationships between small molecules and proteins can be used to support the completion of metabolic networks. Considering the metabolites in StreptomeDB, this approach will find the proteins mainly related to the identified natural products. The presented results and ideas show that there is a need for further interconnection of the research areas of pharmacy, chemistry, molecular biology, and computer science. It is the task of computer scientists to process huge amounts of data in a manageable time, to incorporate new technologies, and to take care for the users' needs. At the same time, computational findings need to be considered and evaluated regularly by specialists from the life science areas. Thus, research will make large progress in the near future.

A. Appendix

A.1. Toxicity-related BestFirst Descriptor Overlaps

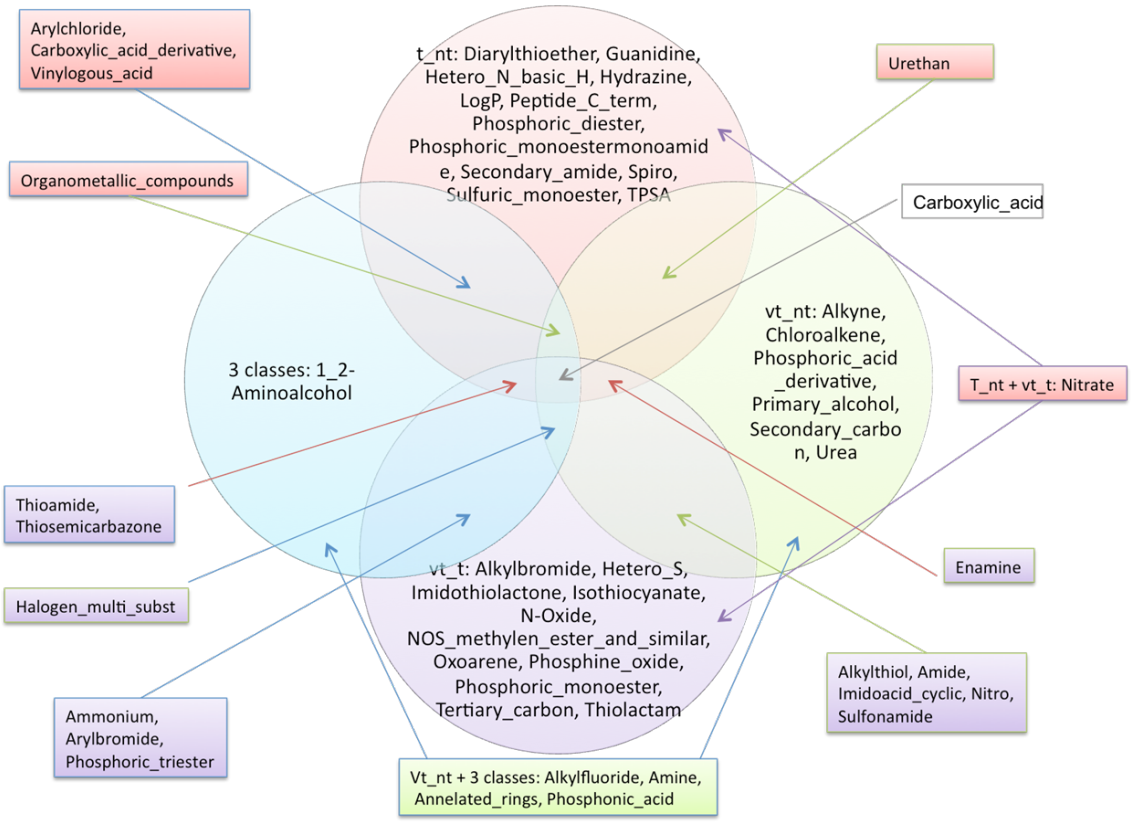


Figure A.1.: BestFirst selection for the data sets with 324 descriptors [51]. The figure is divided into four models, namely three two-class models and one three-class model. The two-class models are abbreviated with t_nt for toxic/non-toxic, vt_nt for very toxic/non-toxic, and vt_t for very toxic/toxic. The descriptors contained in one or more class models are shown directly within the circles or with arrows. The descriptor carboxylic acid is contained in all models.

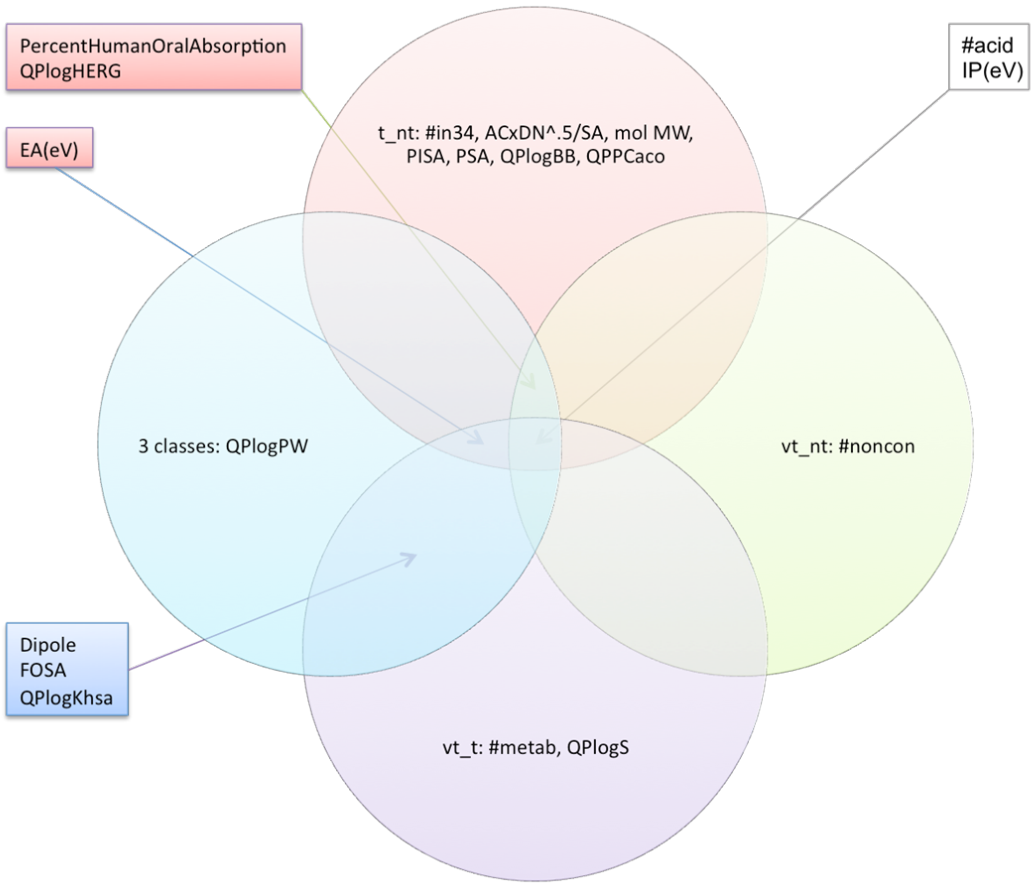


Figure A.2.: BestFirst selection for the data sets with QikProp descriptors [51]. The figure is divided into four models, namely three two-class models and one three-class model. The two-class models are abbreviated with t_nt for toxic/non-toxic, vt_nt for very toxic/non-toxic, and vt_t for very toxic/toxic. The descriptors contained in one or more class models are shown directly within the circles or with arrows. The descriptors #acid and IP(eV) are contained in all models.

A.2. OpenBabel Descriptors

```
1_2-Aminoalcohol:   [OX2H][CX4;!$(C([OX2H])[O,S,#7,#15,F,Cl,Br,I])][CX4;!
                    $(C([N])[O,S,#7,#15])][NX3;!$(NC=[O,S,N])]
Alkene:             [CX3;$([H2]),$([H1][#6]),$(C([#6])[#6])]=[CX3;$([H2]),
                    $([H1][#6]),$(C([#6])[#6])]
Alkylbromide:       [BrX1][CX4]
Alkylfluoride:      [FX1][CX4]
Alkylthiol:         [SX2H][CX4;!$(C([SX2H])~[O,S,#7,#15])]
Alkyne:             [CX2]#[CX2]
Amide:              [CX3;$([R0][#6]),$([H1R0])](=[OX1])[#7X3;$([H2]),$([H1][#6;!
                    $(C=[O,N,S])]),$([#7]([#6;!$(C=[O,N,S])])[#6;!
                    $(C=[O,N,S])])]
Amine:              [NX3+0,NX4+;!$([N]~[!#6]);!$([N]*~[#7,#8,#15,#16])]
Ammonium:           [N+;!$([N]~[!#6]);!$(N=*);!$([N]*~[#7,#8,#15,#16])]
Annelated_rings:    [R;$(*(@*)(@*)@*);!$([R2;$(*(@*)(@*)(@*)@*)])]@[R;
                    $(*(@*)(@*)@*);!$([R2;$(*(@*)(@*)(@*)@*)])]
Arylbromide:        [Br][c]
Arylchloride:       [Cl][c]
Carboxylic_acid:    [CX3;$([R0][#6]),$([H1R0])](=[OX1])[$([OX2H]),
                    $([OX1-])]
Carboxylic_acid_
derivative:         [$([#6X3H0][#6]),$([#6X3H])](=[!#6])[!#6]
Chloroalkene:       [ClX1][CX3]=[CX3]
CH-acidic_strong:   [CX4;!$([H0]);!$(C[!#6;!$([P,S]=O);!$(N(~O)~O)])]([
                    $([CX3]=[O,N,S]),$(C#[N]),$([S,P]=[OX1]),$([NX3]=O),
                    $([NX3+](=O)[O-]);!$(*[S,O,N;H1,H2]);!$([*+0][S,O;X1-])])[
                    $([CX3]=[O,N,S]),$(C#[N]),$([S,P]=[OX1]),$([NX3]=O),
                    $([NX3+](=O)[O-]);!$(*[S,O,N;H1,H2]);!$([*+0][S,O;X1-])]
```

Figure A.3.: OpenBabel descriptors 1/4. First of four parts showing OpenBabel descriptors selected by BestFirst from the the overall amount of 324 descriptors.

```
Diarylthioether:    [c][SX2][c]
Enamine:            [NX3;$([NH2][CX3]),$([NH1]([CX3])[#6]),
                    $([N]([CX3])([#6])[#6]);!
                    $([N]*=[#7,#8,#15,#16])][CX3;$([CH]),$([C][#6])]=[CX3]
Guanidine:          [N;v3X3,v4X4+][CX3](=[N;v3X2,v4X3+])[N;v3X3,v4X4+]
Halogen_multi_
subst:              [F,Cl,Br,I;!$([X1]);!$([X0-])]
Halogen_on_hetero:  [FX1,ClX1,BrX1,IX1][!#6]
Hetero_N_basic_H:   [nX3H1+0]
Hetero_S:           [sX2]
Hydrazine:          [NX3;$([H2]),$([H1][#6]),$([H0]([#6])[#6]);!
                    $(NC=[O,N,S])][NX3;$([H2]),$([H1][#6]),$([H0]([#6])[#6]);!
                    $(NC=[O,N,S])]
Imidoacid_cyclic:   [#6R][#6X3R](=,:[#7X2;$([H1]),$([H0][#6;!$(C=[O,N,S])])])[
                    $([OX2H]),$([OX1-])]
Imidothiolactone:   [#6R][#6X3R](=,:[#7X2;$([H1]),$([H0][#6;!
                    $(C=[O,N,S])])])[SX2][#6;!
                    $(C=[O,N,S])]
Isothiocyanate:     [NX2]=[CX2]=[SX1]
LogP:               Octanol/water partition coefficient
Nitrate:            [$([NX3](=[OX1])(=[OX1])[O;$([X2]),$([X1-])]),
                    $([NX3+]([OX1-])(=[OX1])[O;
                    $([X2]),$([X1-])])]
Nitro:              [$([NX3](=O)=O),$([NX3+](=O)[O-])][!#8]
NOS_methylen_ester_
and_similar:        [NX3v3,SX2,OX2;$(**=[#7,#8,#15,#16])][CX4;!
                    $(C([N,S,O])([N,S,O])[!#6])][NX3v3,SX2,OX2;!
                    $(*C=[#7,#8,#15,#16])]
N-Oxide:            [$([#7+][OX1-]),$([#7v5]=[OX1]);!
                    $([#7](~[O])~[O]);!$([#7]=[#7])]
Organometallic_
compounds:          [!#1;!#5;!#6;!#7;!#8;!#9;!#14;!#15;!#16;!#17;!#33;!#34;
                    !#35;!#52;!#53;!#85]~[#6;!-]
```

Figure A.4.: OpenBabel descriptors 2/4. Second of four parts showing BestFirst selected OpenBabel descriptors. LogP is one of the two selected physicochemical properties beside all the SMARTS patterns.

```
Oxoarene:           [c]=[OX1]
Peptide_C_term:     [NX3;$([N][CX3](=[OX1])[C][NX3,NX4+])]
                    [C][CX3](=[OX1])[OX2H,OX1-]
Phosphine_oxide:    [PX4;$([H3]=[OX1]),$([H2](=[OX1])[#6]),
                    $([H1](=[OX1])([#6])[#6]),$([H0](=[OX1])([#6])([#6])[#6])]
Phosphonic_acid:    [PX4;$([H1]),$([H0][#6])](=[OX1])([$([OX2H]),
                    $([OX1-])])[$([OX2H]),$([OX1-])]
Phosphoric_acid_
derivative:         [PX4D4](=[!#6])([!#6])([!#6])[!#6]
Phosphoric_diester: [PX4D4](=[OX1])([$([OX2H]),$([OX1-])])([OX2][#6;!
                    $(C=[O,N,S])])[OX2][#6;!$(C=[O,N,S])]
Phosphoric_
monoestermonoamide: [PX4D4](=[OX1])([$([OX2H]),$([OX1-])])([OX2][#6;!
                    $(C=[O,N,S])])[#7X3;$([H2]),$([H1][#6;!$(C=[O,N,S])]),
                    $([#7]([#6;!$(C=[O,N,S])])[#6;!$(C=[O,N,S])])]
Phosphoric_
monoester:          [PX4D4](=[OX1])([$([OX2H]),$([OX1-])])([$([OX2H]),
                    $([OX1-])])[OX2][#6;!$(C=[O,N,S])]
Phosphoric_
triester:           [PX4D4](=[OX1])([OX2][#6;!$(C=[O,N,S])])([OX2][#6;!
                    $(C=[O,N,S])])[OX2][#6;!$(C=[O,N,S])]
Primary_alcohol:    [OX2H][CX4H2;!$(C([OX2H])[O,S,#7,#15])]
Secondary_amide:    [CX3;$([R0][#6]),$([H1R0])](=[OX1])[#7X3H1][#6;!
                    $(C=[O,N,S])]
Secondary_carbon:   [CX4H2]([#6])[#6]
Spiro:              [D4R;$(*(@*)(@*)(@*)@*)]
Sulfonamide:        [SX4;$([H1]),$([H0][#6])](=[OX1])(=[OX1])[#7X3;$([H2]),
                    $([H1][#6;!$(C=[O,N,S])]),$([#7]([#6;!
                    $(C=[O,N,S])])[#6;!$(C=[O,N,S])])]
```

Figure A.5.: OpenBabel descriptors 3/4. Third of four parts showing OpenBabel descriptors based on the BestFirst search.

```
Sulfonic_acid:      [SX4;$([H1]),$([H0][#6])](=[OX1])(=[OX1])[$([OX2H]),
                    $([OX1-])]
Sulfuric_monoester: [SX4](=[OX1])(=[OX1])([$([OX2H]),$([OX1-])])[OX2][#6;!
                    $(C=[O,N,S])]
Tertiary_carbon:    [CX4H1]([#6])([#6])[#6]
Thioamide:          [$([CX3;!R][#6]),$([CX3H;!R])](=[SX1])[#7X3;$([H2]),
                    $([H1][#6;!$(C=[O,N,S])]),$([#7]([#6;!$(C=[O,N,S])])[#6;!
                    $(C=[O,N,S])])]
Thiolactam:         [#6R][#6X3R](=[SX1])[#7X3;$([H1][#6;!$(C=[O,N,S])]),
                    $([H0]([#6;!$(C=[O,N,S])])[#6;!$(C=[O,N,S])])]
Thiosemicarbazone:  [#7X2](=[#6])[#7X3][#6X3]([#7X3;!$([#7][#7])])=[SX1]
TPSA:               Topological polar surface area
Urea:               [#7X3;!$([#7][!#6])][#6X3](=[OX1])[#7X3;!$([#7][!#6])]
Urethan:            [#7X3][#6](=[OX1])[#8X2][#6]
Vinylogous_acid:    [#6X3](=[OX1])[#6X3]=,:[#6X3][$([OX2H]),$([OX1-])]
```

Figure A.6.: OpenBabel descriptors 4/4. Remaining part showing BestFirst selected OpenBabel descriptors. TPSA represents the second BestFirst selected physico-chemical property.

All SMARTS patterns can be further evaluated in OpenBabel and with the SMARTSviewer as described in Section 2.3.2.

A.3. QikProp Descriptors

All ranges presented within the QikProp descriptor set are related to what was found in 95 % of all known drugs (Section 3.3). However, the data set, this investigation is based on, is not further refined[1].

```
#acid:          Number of carboxylic acid groups (range: 0-1).
ACxDN^.5/SA:    Index of cohesive interaction in solids. This term represents
                the relationship ( accptHB ( sqrt(donorHB )) ) / ( SA ) ;
                [Jorgensen, W. L.; Duffy, E. M. Prediction of Drug Solubility
                from Monte Carlo Simulations. Bioorg. Med. Chem. Lett. 2000, 10,
                1155-1158.]
dipole[†]:        Computed dipole moment of the molecule. (range: 1.0-12.5)
EA(eV)[†]:        PM3 calculated electron affinity (negative of LUMO energy)
                (range: -0.9-1.7).
FOSA:           Hydrophobic component of the SASA (saturated carbon and attached
                hydrogen, range: 0-750.0).
IP(ev)[†]:        PM3 calculated ionization potential (negative of HOMO energy,
                (range: 7.9-10.5).
#in34:          Number of atoms in 3- or 4-membered rings
```

Figure A.7.: QikProp descriptors 1/3. First of three parts showing QikProp descriptors selected by BestFirst from the the overall amount of 51 descriptors.

```
#metab[‡]:      Number of likely metabolic reactions.  The following metabolic
                reactions contribute to this descriptor (range: 1-8):
                    aromatic OH oxidation
                    enol oxidation
                    benzylic-like H -> alcohol
                    allylic H -> alcohol
                    secondary alcohol -> ketone
                    primary alcohol -> acid
                    tertiary alcohol E1 or SN1
                    amine dealkylation
                    ether dealkylation
                    pyridine C2 hydroxylation
                    aniline NH -> NOH or NCOR
                    low IP - easily oxidized
                    alpha hydroxylation of cyclic ether
                    sulfoxide-> sulfone
                    alpha hydroxylation of carbonyl
                    alpha, beta dehydrogenation at carbonyl
                    thiol SH-> SSR, SR
                    para hydroxylation of aryl
                    aryl sulfide -> S=O
                    reduction of aryl nitro to amine
                    oxidative deamination of primary amine.
```

Figure A.8.: QikProp descriptors 2/3. This descriptor assigns metabolic reactions and functions to a given molecule.

[1] http://helixweb.nih.gov/schrodinger-2013.3-docs/qikprop/qikprop_user_manual.pdf

```
mol_MW:             Molecular weight of the molecule (range:130.0-725.0).
#noncon:            number of ring atoms not able to form conjugated
                    aromatic systems (e.g. sp^3 C).
PercentHumanOral
Absorption:         Predicted human oral absorption on 0 to 100% scale. The
                    prediction is based on a quantitative multiple linear
                    regression model. This property usually correlates well
                    with HumanOralAbsorption, as both measure the same property
                    (range: >80% is high, <25% is poor).
PISA:               Pi (carbon and attached hydrogen) component of the SASA
                    (SASA: SASA Total solvent accessible surface area (SASA)
                    in square angstroms using a probe with a 1.4 Å radius.)
                    (range: 0.0-450.0).
PSA:                Van der Waals surface area of polar nitrogen and oxygen
                    atoms and carbonyl carbon atoms (range: 7.0-200.0).
QPlogBB:            Predicted brain/blood partition coefficient. Note:
                    QikProp predictions are for orally delivered drugs so,
                    for example, dopamine and serotonin are CNS negative because
                    they are too polar to cross the blood-brain barrier
                    (range: -3.0-1.2).
QPPCaco:            Predicted apparent Caco-2 cell permeability in nm/sec.
                    Caco-2 cells are a model for the gut-blood barrier.
                    QikProp predictions are for non-active transport
                    (range: <25 poor, >500 great).
QPlogHERG:          Predicted IC_50 value for blockage of HERG K^+ channels
                    (range: concern below -5).
QPlogKhsa:          Prediction of binding to human serum albumin
                    (range: -1.5-1.5).
QPlogPw:            Predicted water/gas partition coefficient (range: 4.0-45.0).
QPlogS:             Predicted aqueous solubility, log S. S in mol dm -3 is the
                    concentration of the solute in a saturated solution that is
                    in equilibrium with the crystalline solid (range: -6.5-0.5).
```

Figure A.9.: QikProp descriptors 3/3. Last of three parts showing BestFirst selected QikProp descriptors.

A.4. WEKA Toxicity Decision Tree

```
Carboxylic_acid <= 0
|  Salt <= 0
|  |  Primary_alcohol <= 0
|  |  |  Annelated_rings <= 0: toxic (760.0/354.0)
|  |  |  Annelated_rings > 0
|  |  |  |  Annelated_rings <= 3
|  |  |  |  |  Halogen_on_hetero <= 0
|  |  |  |  |  |  Annelated_rings <= 1: toxic (156.0/78.0)
|  |  |  |  |  |  Annelated_rings > 1
|  |  |  |  |  |  |  Tertiary_carbon <= 2
|  |  |  |  |  |  |  |  Tertiary_carbon <= 1
|  |  |  |  |  |  |  |  |  Alkene <= 0: toxic (39.0/16.0)
|  |  |  |  |  |  |  |  |  Alkene > 0: very toxic (12.0/4.0)
|  |  |  |  |  |  |  |  Tertiary_carbon > 1: very toxic (9.0/1.0)
|  |  |  |  |  |  |  Tertiary_carbon > 2: toxic (4.0)
|  |  |  |  |  Halogen_on_hetero > 0
|  |  |  |  |  |  Hetero_N_basic_no_H <= 0: very toxic (35.0/11.0)
|  |  |  |  |  |  Hetero_N_basic_no_H > 0: toxic (10.0/1.0)
|  |  |  |  Annelated_rings > 3: very toxic (38.0/6.0)
|  |  Primary_alcohol > 0
|  |  |  Tertiary_carbon <= 1
|  |  |  |  Primary_alcohol <= 1
|  |  |  |  |  Annelated_rings <= 0
|  |  |  |  |  |  Halogen_on_hetero <= 0: nontoxic (49.0/18.0)
|  |  |  |  |  |  Halogen_on_hetero > 0: toxic (12.0/5.0)
|  |  |  |  |  Annelated_rings > 0: toxic (20.0/9.0)
|  |  |  |  Primary_alcohol > 1: nontoxic (38.0/6.0)
|  |  |  Tertiary_carbon > 1: very toxic (5.0/1.0)
|  Salt > 0
|  |  Sulfonic_acid <= 0
|  |  |  Hetero_N_basic_no_H <= 0: very toxic (655.0/137.0)
|  |  |  Hetero_N_basic_no_H > 0
|  |  |  |  Hetero_N_basic_no_H <= 2
|  |  |  |  |  Primary_alcohol <= 0: very toxic (13.0/3.0)
|  |  |  |  |  Primary_alcohol > 0: nontoxic (2.0/1.0)
|  |  |  |  Hetero_N_basic_no_H > 2: nontoxic (8.0/1.0)
|  |  Sulfonic_acid > 0
|  |  |  Salt <= 3: toxic (23.0/11.0)
|  |  |  Salt > 3
|  |  |  |  Sulfonic_acid <= 2: very toxic (4.0)
|  |  |  |  Sulfonic_acid > 2: nontoxic (6.0/1.0)
Carboxylic_acid > 0
|  Alkene <= 0
|  |  Carboxylic_acid <= 1
|  |  |  Annelated_rings <= 1: nontoxic (467.0/151.0)
|  |  |  Annelated_rings > 1
|  |  |  |  Hetero_N_basic_no_H <= 1: toxic (29.0/12.0)
|  |  |  |  Hetero_N_basic_no_H > 1: very toxic (3.0)
|  |  Carboxylic_acid > 1: nontoxic (170.0/36.0)
|  Alkene > 0
|  |  Annelated_rings <= 0: toxic (37.0/14.0)
|  |  Annelated_rings > 0: nontoxic (9.0)
```

Figure A.10.: Three-class decision tree J4.8 based on 10 descriptors [51]. In contrast to Figure 3.31, this is the complete decision tree based on 10 descriptors.

Bibliography

[1] B.A. Grüning, C. Senger, et al. Compounds In Literature (CIL): screening for compounds and relatives in PubMed. *Bioinformatics*, 27, 2011.

[2] C. Senger, B.A. Grüning, et al. Mining and evaluation of molecular relationships in literature. *Bioinformatics*, 28, 2012.

[3] E. Bolton et al. *Chapter 12 - PubChem: Integrated Platform of Small Molecules and Biological Activities.* In Wheeler RA and Spellmeyer DC, *Annual Reports in Computational Chemistry*, Oxford, 2008.

[4] UniProt Consortium. UniProt: a hub for protein information. *Nucleic Acids Res*, 43, 2015.

[5] S. Ekins. *Computational Toxicology - Risk Assessment for Pharmacetical and Environmental Chemicals.* In Wiley Series on Technologies for the Pharmaceutical Industry, 2007.

[6] S.F. Altschul et al. Basic local alignment search tool. *J Mol Biol*, 215, 1990.

[7] D. Rebholz-Schuhmann et al. Text processing through Web services: calling Whatizit. *Bioinformatics*, 24, 2008.

[8] R. Khare et al. *Accessing biomedical literature in the current information landscape.* In Biomedical Literature Mining, Springer Protocols, *Methods Mol Biol*, 2014.

[9] D. Tikk et al. A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature. *PLoS Comput Biol*, 6, 2011.

[10] M. Kuhn et al. STITCH 4: integration of protein-chemical interactions with user data. *Nucleic Acids Res*, 42, 2014.

[11] L. Tari et al. Discovering drug-drug interactions: a text-mining and reasoning approach based on properties of drug metabolism. *Bioinformatics*, 26, 2010.

[12] I. Segura-Bedmar et al. Using a shallow linguistic kernel for drug-drug interaction extraction. *J Biomed Inform*, 44, 2011.

[13] F. Rinaldi et al. OntoGene web services for biomedical text mining. *BMC Bioinformatics*, 15, 2014.

[14] Gene Ontology Consortium. Gene Ontology Consortium: going forward. *Nucleic Acids Res*, 43, 2015.

[15] R. Merkel. *Bioinformatik: Grundlagen, Algorithmen, Anwendungen*. Wiley-VCH, 2015.

[16] Bickerton G.R. et al. Quantifying the chemical beauty of drugs. *Nat Chem*, 4, 2012.

[17] L.J. Jensen et al. Literature mining for the biologist: from information retrieval to biological discovery. *Nat Rev Genet*, 7, 2006.

[18] J.R. Anderson. *Kognitive Psychologie*. Spektrum Akademischer Verlag, 2. Aufl., 1996.

[19] G. Schneider and K.-H. Baringhaus. *Molecular Design, Concepts and Applications*. Wiley-VCH, 2008.

[20] K. Döring. Implementation and application of machine learning approaches for toxicity prediction. http://bioinformatics.charite.de/main/content/theses.php, 2009. Bachelor Thesis.

[21] D. Schuster et al. Why drugs fail - A study on side effects in new chemical entities. *Curr Pharm Des*, 11, 2005.

[22] J. Drews. *Die verspielte Zukunft, Wohin geht die Arzneimittelforschung?* Springer, 1998.

[23] H.-J. Böhm. *Wirkstoffdesign*. Spektrum Akademischer Verlag, 2002.

[24] G. Klebe. *Wirkstoffdesign, Entwurf und Wirkung von Arzneistoffen*. Spektrum Akademischer Verlag, 2009.

[25] I.H. Witten et al. *Data Mining: Practical Machine Learning Tools and Techniques*. In Morgan Kaufmann Publishers, Elsevier, 2011.

[26] C.P. Koch et al. Scrutinizing MHC-I Binding Peptides and Their Limits of Variation. *PLoS Comput Biol*, 9, 2013.

[27] W.S. Sanders et al. Prediction of Cell Penetrating Peptides by Support Vector Machines. *PLoS Comput Biol*, 7, 2011.

[28] S. Hayata et al. All-atom 3D structure prediction of transmembrane β-barrel proteins from sequences. *Proc Natl Acad Sci U S A*, 112, 2015.

[29] X. Lucas, C. Senger, et al. StreptomeDB: a resource for natural compounds isolated from Streptomyces species. *Nucleic Acids Res*, 41, 2013.

[30] M. Ventura et al. Genomics of Actinobacteria: tracing the evolutionary history of an ancient phylum. *Microbiol Mol Biol Rev*, 71, 2007.

[31] D.A. Hopwood et al. *Streptomyces in Nature and Medicine: The Antibiotic Makers*. Oxford Univ Pr, 2007.

[32] M.O. Griffin et al. StreptomeDB: a resource for natural compounds isolated from Streptomyces species. *Am J Physiol Cell Physiol*, 299, 2007.

[33] J.R. Woodworth et al. Single-dose pharmacokinetics and antibacterial activity of daptomycin, a new lipopeptide antibiotic, in healthy volunteers. *Antimicrob Agents Chemother*, 36, 1992.

[34] N. Nathan et al. Ceftriaxone as effective as long-acting chloramphenicol in short-course treatment of meningococcal meningitis during epidemics: a randomised non-inferiority study. *Lancet*, 366, 2005.

[35] D. Jones et al. CONTROL OF GRAM-NEGATIVE BACTERIA IN EXPERIMENTAL ANIMALS BY STREPTOMYCIN. *Science*, 100, 1944.

[36] M.S. Butler and A.D. Buss. Natural products–the future scaffolds for novel antibiotics? *Biochem Pharmacol*, 71, 2006.

[37] A. Kling et al. Antibiotics. Targeting DnaN for tuberculosis therapy using novel griselimycins. *Science*, 348, 2015.

[38] M.S. Butler and A.D. Buss. The effects of some avermectins on bovine carbonic anhydrase enzyme. *J Enzyme Inhib Med Chem*, 2015.

[39] E.I. Graziani. Recent advances in the chemistry, biosynthesis and pharmacology of rapamycin analogs. *Nat Prod Rep*, 26, 2009.

[40] T. Bai et al. Operon for biosynthesis of lipstatin, the Beta-lactone inhibitor of human pancreatic lipase. *Appl Environ Microbiol*, 80, 2014.

[41] D. Dhakal and J.K. Sohng. Commentary: Toward a new focus in antibiotic and drug discovery from the Streptomyces arsenal. *Front Microbiol*, 6, 2015.

[42] D. Klementz, K. Döring, et al. StreptomeDB 2.0 - Centralized knowledge of secondary metabolites produced by streptomycetes. Nucleic Acids Res, 2016. Under submission.

[43] M. Huang et al. GeneTUKit: a software for document-level gene normalization. *Bioinformatics*, 27, 2011.

[44] R. Leaman and G. Gonzalez. BANNER: an executable survey of advances in biomedical named entity recognition. *Pac Symp Biocomput*, 2008.

[45] D Maglott et al. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 35, 2007.

[46] B. Settles. ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21, 2005.

[47] E. Abbasian. Extraction of Compound-Protein Interactions from PubMed Using the All-Paths Graph Kernel. http://www.pharmaceutical-bioinformatics.de/main/theses, 2015. Master Thesis.

[48] J. Hakenberg et al. The GNAT library for local and remote gene mention normalization. *Bioinformatics*, 27, 2011.

[49] K.M Hettne et al. A dictionary to identify small molecules and drugs in free text. *Bioinformatics*, 25, 2009.

[50] R. Kabilo et al. A realistic assessment of methods for extracting gene/protein interactions from free text. *BMC Bioinformatics*, 10, 2009.

[51] L.M. Grüger. Identification of Molecular Descriptors for Toxicity Prediction of Small Molecules. http://www.pharmaceutical-bioinformatics.de/main/theses, 2013. Bachelor Thesis.

[52] T. Fawcett. An introduction to ROC analysis. *Pattern Recognit Lett*, 27, 2006.

[53] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classiers. *In Proc. Eighteenth Internat. Conf. on Machine Learning*, 2001.

[54] A. Airola et al. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9, 2008.

[55] G. Görz. *Handbuch der künstlichen Intelligenz*. Oldenbourg Wissenschaftsverlag GmbH, München, 4th edition, 2003.

[56] R. Bunescu and R. Mooney. Subsequence Kernels for Relation Extraction. *In Advances in Neural Information Processing Systems 18, MIT Press*, 2006.

[57] T. Pahikkala et al. Graph Kernels versus Graph Representations: a Case Study in Parse Ranking. *Proceedings of the Fourth Workshop on Mining and Learning with Graphs*, 2006.

[58] R. Rojas. *Neural Networks - A Systematic Introduction.* Springer-Verlag, Berlin, 2011.

[59] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev*, 65, 2008.

[60] S. Schneider et al. Peptide design by artificial neural networks and computer-based evolutionary search. *Proc Natl Acad Sci U S A*, 95, 1998.

[61] R. Herbrich et al. Learning a Preference Relation in Information Retrieval. *In Proceedings Workshop Text Categorization and Machine Learning, International Conference on Machine Learning*, 1998.

[62] N. Christianini and J.S. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[63] C Giuliano et al. Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature. *In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, 2006.

[64] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[65] T. Gärtner et al. On graph kernels: Hardness results and efficient alternatives. *In Learning Theory and Kernel Machines, Lecture Notes in Computer Science*, 2777, 2003.

[66] R. Rifkin et al. Regularized Least-Squares Classification. *Advances in Learning Theory: Methods, Model and Applications NATO Science Series III: Computer and Systems Sciences*, 190, 2003.

[67] Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[68] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[69] S. Aksoy and R.M. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognit Lett*, 22, 2001.

[70] M. Hall. Feature Selection for Discrete and Numeric Class Machine Learning. *In Machine Learning. Proc. Seventeenth International conference on Machine Learning*, 2000.

[71] A. Hyvärinen et al. *Natural Image Statistics - A probabilistic approach to early computational vision.* In Computational Imaging, Springer-Verlag, 2009.

[72] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, 1936.

[73] J. Gasteiger and T. Engel. *Chemoinformatics: A Textbook.* In Wiley-VCH, 2003.

[74] F.-X. Reichl. *Taschenatlas Toxikologie.* Thieme, 2009.

[75] K. Döring et al. PubMed2Go: A Framework for Developing Text Mining Applications. BMC Bioinformatics, 2015. Under submission.

[76] R. Rak et al. Processing biological literature with customizable Web services supporting interoperable formats. *Database (Oxford)*, 2003.

[77] D. Ferruci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng*, 10, 2004.

[78] H. Cunningham et al. *Processing with Gate.* Gateway Press, Murphys, CA., 2011.

[79] D.C. Comeau et al. BioC: a minimalist approach to interoperability for biomedical text processing. *Database (Oxford)*, 2013.

[80] Y. Kano et al. U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*, 25, 2009.

[81] R. Khare et al. tmBioC: improving interoperability of text-mining tools with BioC. *Database (Oxford)*, 2014.

[82] C.-H. Wei et al. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Res*, 41, 2013.

[83] D.E. Oliver et al. Tools for loading MEDLINE into a local relational database. *BMC Bioinformatics*, 5, 2004.

[84] D. Yoo et al. *UNIT 9.7 PubSearch and PubFetch: a simple management system for semiautomated retrieval and annotation of biological information from the literature.* In John Wiley & Sons, Inc., *Curr Protoc Bioinformatics*, 2006.

[85] D.C. Comeau et al. Natural language processing pipelines to annotate BioC collections with an application to the NCBI disease corpus. *Database (Oxford)*, 2014.

[86] F. Rinaldi and H. Marques. PyBioC: a Python implementation of the BioC core. *Proceedings of the Fourth BioCreative Challenge Evaluation Workshop*, 1, 2013.

[87] D.C. Comeau et al. BioC interoperability track overview. *Database (Oxford)*, 2014.

[88] I. Garrido-Laguna and M. Hidalgo. Pancreatic cancer: from state-of-the-art treatments to promising novel therapies. *Nat Rev Clin Oncol*, 12, 2015.

[89] V. Law et al. DrugBank 4.0: shedding new light on drug metabolism. *Nucleic Acids Res*, 2014.

[90] D.P. Ryan et al. Pancreatic adenocarcinoma. *N Engl J Med*, 371, 2014.

[91] G. Zimmerman et al. Small molecule inhibition of the KRAS-PDEδ interaction impairs oncogenic KRAS signalling. *Nature*, 497, 2013.

[92] A. Spirin and L.A. Mirny. Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A*, 100, 2012.

[93] A. Gaulton et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res*, 40, 2012.

[94] J. Saric et al. Extraction of regulatory gene/protein networks from Medline. *Bioinformatics*, 22, 2013.

[95] A. Franceschini et al. STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41, 2013.

[96] F. Rinaldi et al. Using the OntoGene pipeline for the triage task of BioCreative 2012. *Database (Oxford)*, 41, 2013.

[97] M. Becer. Automatisierte Identifizierung funktioneller Beziehungen zwischen Kleinmolekülen und Proteinen in Texten. http://www.pharmaceutical-bioinformatics.de/main/theses, 2014. Bachelor Thesis.

[98] Y. Tsuruoka and J. Tsujii. Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data. *Proceedings of HLT/EMNLP 2005*, 2005.

[99] N. Kang et al. Comparing and combining chunkers of biomedical text. *J Biomed Inform*, 23, 2011.

[100] E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005.

[101] M.F. Gatnik and A. Worth. Review of Software Tools for Toxicity Prediction. *JRC Scientific and Technical Reports*, 2010.

[102] N.M. O'Bolyle et al. Open Babel: An open chemical toolbox. *J Cheminform*, 3, 2011.

[103] X. Lucas et al. ChemicalToolBoX and its application on the study of the drug like and purchasable space. J Cheminform, 2014. Poster presentation.

[104] K. Fundel et al. RelEx–relation extraction using dependency parse trees. *Bioinformatics*, 23, 2007.

[105] T. Rocktäschel et al. Discovering relations between indirectly connected biomedical concepts. *Bioinformatics*, 28, 2012.

[106] S. Pyysalo et al. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8, 2007.

[107] D. Weissenborn et al. Discovering relations between indirectly connected biomedical concepts. *J Biomed Semantics*, 6, 2015.

[108] L. Yeganova et al. Retro: concept-based clustering of biomedical topical sets. *Bioinformatics*, 30, 2014.

[109] R. Khare et al. Crowdsourcing in biomedicine: challenges and opportunities. *Brief Bioinform*, 2015.

[110] H. Geppert et al. Support-Vector-Machine-Based Ranking Significantly Improves the Effectiveness of Similarity Searching Using 2D Fingerprints and Multiple Reference Compounds. *J Chem Inf Model*, 48, 2008.

[111] L. Xue and J. Bajorath. Molecular Descriptors for Effective Classification of Biologically Active Compounds Based on Principal Component Analysis Identified by a Genetic Algorithm. *J Chem Inf Comput Sci*, 40, 2000.

[112] E. Byvatov et al. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *J Chem Inf Comput Sci*, 43, 2003.

[113] R. Todeschini and V. Consonni. *Molecular Descriptors for Chemoinformatics: Volume I: Alphabetical Listing / Volume II: Appendices, References.* In Wiley-VCH, *Methods and Principles in Medicinal Chemistry*, 2009.

[114] M. Schmuker et al. SOMMER: self-organising maps for education and research. *J Mol Model*, 13, 2007.

[115] D.P. Hristozov et al. Virtual screening applications: a study of ligand-based methods and different structure representations in four different scenarios. *J Comput Aided Mol Des*, 21, 2007.

[116] M. Nencki et al. *Über den Einfluss der Carboxylgruppe auf die Toxische Wirkung aromatischer Substanzen.* In Springer, *Archiv für experimentelle Pathologie und Pharmakologie*, 1892.

[117] S.E. Manahan. *Fundamentals of Environmental Chemistry.* In CRC Press, Lewis Publishers, 2011.

[118] R. Liu et al. 2D SMARTCyp reactivity-based site of metabolism prediction for major drug-metabolizing cytochrome P450 enzymes. *J Chem Inf Model*, 52, 2012.

[119] D.M. Jessop et al. OSCAR4: a flexible architecture for chemical text-mining. *J Cheminform*, 3, 2011.

[120] I. Thiele et al. A community-driven global reconstruction of human metabolism. *Nat Biotechnol*, 2013.

[121] O. Hagan et al. A 'rule of 0.5' for the metabolite-likeness of approved pharmaceutical drugs. *Metabolomics*, 2015.

[122] X. Lucas et al. The purchasable chemical space: a detailed picture. *J Chem Inf Model*, 2015.

[123] C.A. Lipinski et al. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv Drug Deliv Rev*, 2001.

[124] F. Ntie-Kang. An in silico evaluation of the ADMET profile of the StreptomeDB database. *Springerplus*, 2013.

Acknowledgements

First of all, I want to thank my supervisor Stefan Günther. The time in your working group truly broadened my mind and I am proud of what I could achieve here in my working time with you. Thank you for this opportunity.

Thank you Prof. Wrede, for having me led to the field of machine learning and for being my supervisor, even in the time there was no thesis I wanted to write.

I want to thank my family, especially my mother. I cannot remember a single moment in which you would not have been there for me. Thank you father, thank you Mimi, and thank you all lovely members in my family. I missed you many times.

With all my heart, I want to thank you for everything, Evelyn, for your love and your support. I also want to express my gratitude to your family and your friends, you are amazing.

A few people in and close to my home town showed me the value of being friends. Thank you all for enriching my life during my time at home and also when I was far away.

I want to thank a few people here in Freiburg and I guess the persons know who I mean. My second home town would not have been so lovely without you.

Many kind and interesting people passed my way during the PhD time and I cannot thank all of them within the few lines that are left in this thesis.

Thank you guys in my working group and all related persons I could laugh, work, discuss, and celebrate with. Thanks to the students who I was allowed to supervise, I think we all learned a lot. The time was tough, but full of fun as well. The world is small and in particular Freiburg. I can assure that we will see each other, again.

This is a fairly good university. Thanks to all the people who supported me.