Albert-Ludwigs-Universität Freiburg

**Automatisierte Identifizierung von Wirkstoffen in wissenschaftlicher Literatur mithilfe maschineller Lernverfahren**

Bachelorarbeit

*Pharmazeutische Wissenschaften*

Manuel Dorer

Fakultät für Chemie und Pharmazie,

Institut für Pharmazeutische Wissenschaften,

Pharmazeutische Bioinformatik

vorgelegt von

Manuel Dorer

am XX.XX.2021, Freiburg

Erstgutachter: Prof. Dr. Stefan Günther

Pharmazeutische Bioinformatik,

Albert-Ludwigs-Universität Freiburg

Zweitgutachter: Jun.-Prof. Dr. Jennifer Andexer,

Pharmazeutische und Medizinische Chemie,

Albert-Ludwigs-Universität Freiburg

Betreuer: Ammar Qaseem,

Pharmazeutische Bioinformatik,

Albert-Ludwigs-Universität Freiburg



Eigenständigkeitserklärung

Name des Studenten: Manuel Dorer

Matrikelnummer: 4531053

*ERKLÄRUNG*

*zur Abgabe der Bachelor-Arbeit   
im Studiengang B.Sc. Pharmazeutische Wissenschaften*

Hiermit versichere ich, dass ich die Bachelorarbeit selbstständig verfasst und keine andere als die von mir angegebenen Quellen und Hilfsmittel benutzt habe. Alle Zitate sind gekennzeichnet und alle Abbildungen enthalten nur die originalen Daten und sind in keinem Fall inhaltsverändernder Bildbearbeitung unterzogen worden. Die abgegebene schriftliche und elektronische Fassung sind identisch. Weiterhin versichere ich, dass die Arbeit noch nicht anderweitig als Bachelorarbeit eingereicht wurde.

............................................ ........................................................

Ort/Datum Unterschrift des Studenten

# Abstract

In recent years, deep learning has garnered tremendous success in a variety of application

domains. This new field of machine learning has been growing rapidly and has been applied to

most traditional application domains, as well as some new areas that present more opportunities.

Different methods have been proposed based on different categories of learning, including

supervised, semi-supervised, and un-supervised learning. Experimental results show state-of-theart

performance using deep learning when compared to traditional machine learning approaches in

the fields of image processing, computer vision, speech recognition, machine translation, art,

medical imaging, medical information processing, robotics and control, bioinformatics, natural

language processing, cybersecurity, and many others. This survey presents a brief survey on the

advances that have occurred in the area of Deep Learning (DL), starting with the Deep Neural

Network (DNN). The survey goes on to cover Convolutional Neural Network (CNN), Recurrent

Neural Network (RNN), including Long Short-Term Memory (LSTM) and Gated Recurrent Units

(GRU), Auto-Encoder (AE), Deep Belief Network (DBN), Generative Adversarial Network (GAN),

and Deep Reinforcement Learning (DRL). Additionally, we have discussed recent developments,

such as advanced variant DL techniques based on these DL approaches. This work considers most

of the papers published after 2012 from when the history of deep learning began. Furthermore, DL

approaches that have been explored and evaluated in different application domains are also

included in this survey. We also included recently developed frameworks, SDKs, and benchmark

datasets that are used for implementing and evaluating deep learning approaches. There are some

surveys that have been published on DL using neural networks and a survey on Reinforcement

Learning (RL). However, those papers have not discussed individual advanced techniques for

training large-scale deep learning models and the recently developed method of generative models.

 Heutzutage sind noch viele Datensätze von Hand zusammengestellt

# **Inhaltsverzeichnis**

[Abstract i](#_Toc67012547)

[Inhaltsverzeichnis ii](#_Toc67012548)

[Abbildungsverzeichnis iii](#_Toc67012549)

[Tabellenverzeichnis iv](#_Toc67012550)

[Abkürzungsverzeichnis v](#_Toc67012551)

[1 Einleitung 1](#_Toc67012552)

[1.1 Begriffe aus dem Feld der KI 1](#_Toc67012553)

[1.1.1 KI 1](#_Toc67012554)

[1.1.2 maschinelles Lernen 2](#_Toc67012555)

[1.2 Wirkstoffsuche im Zeitalter von Big Data 11](#_Toc67012556)

[1.2.1 klassischer Weg 11](#_Toc67012557)

[1.2.2 virtuelles Screening 11](#_Toc67012558)

[1.3 CPI Text Mining 12](#_Toc67012559)

[1.3.1 Historie von RE Text Mining mit Blick auf CPI Text Mining 13](#_Toc67012560)

[2 Zielsetzung der Arbeit 17](#_Toc67012561)

[3 Material und Methoden 18](#_Toc67012562)

[3.1 Grundlagen 18](#_Toc67012563)

[3.1.1 Systemeigenschaften 18](#_Toc67012564)

[3.1.2 Verwendete Software 18](#_Toc67012565)

[3.1.3 Verwendeter Datensatz 19](#_Toc67012566)

[3.1.4 Parameter zur Bewertung von Modellen 19](#_Toc67012567)

[3.2 Datensatzpräparation 21](#_Toc67012568)

[3.3 10-fache Kreuzvalidierung ~~für den Modellvergleich~~ 22](#_Toc67012569)

[3.3.1 -fache Kreuzvalidierung 22](#_Toc67012570)

[3.3.2 Splitten des Datensatzes für 10-fache Kreuzvalidierung 22](#_Toc67012571)

[3.3.3 Durchführung der 10-fache Kreuzvalidierung 24](#_Toc67012572)

[3.3.4 Reproduzieren der Klassifizierungsergebnisse der Kern-Methoden 24](#_Toc67012573)

[3.4 ~~Untersuchung des~~ Einfluss der Trainingsdatensatzgröße 25](#_Toc67012574)

[3.4.1 splitten des DS 26](#_Toc67012575)

[3.4.2 Durchführung 26](#_Toc67012576)

[4 Ergebnisse 25](#_Toc67012577)

[4.1 Kreuzvalidierung für Modellvergleich 25](#_Toc67012578)

[5 Diskussion und Ausblick 25](#_Toc67012579)

[6 Zusammenfassung 27](#_Toc67012580)

[7 Literaturverzeichnis 28](#_Toc67012581)

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

# Einleitung

Am 15. März 2016 stand es fest: AlphaGo besiegt Lee Sedol, den damaligen stärksten Go Spieler der Welt, in einer Partie mit 4:1 [1]. Doch was hat dieses Ereignis mit autonom fahrenden Autos, der Wettervorhersage oder Spracherkennung zu tun?

In allen genannten Bereichen wird zunehmend mit künstlicher Intelligenz (KI) gearbeitet, bzw. gibt es immer mehr Ansätze, welche auf einer KI basieren [2] [3] [4] [5]. KIs kommen häufig dort zum Einsatz, wo große Datenmengen anfallen und zeitnah bewertet werden müssen. Dabei erkennen KIs Muster in den Daten und können Vorhersagen über die weitere Entwicklung treffen, was für das Fällen von Entscheidungen von großer Bedeutung ist. Nicht zuletzt haben KIs auch auf vielfältige Art und Weise ihren Weg in die naturwissenschaftliche Forschung gefunden. Anwendungsgebiete sind z.B. das Stellen von Hautkrebsdiagnosen in der Medizin [6], das Optimieren von Molekülsimulationen für die Werkstoffforschung in der Physik [7], oder generell das Extrahieren von Wissen aus Daten. Besonders letzteres gewinnt im Zeitalter von Big Data immer mehr an Relevanz und soll in dieser Bachelorarbeit im Mittelpunkt stehen.

## Begriffe aus dem Feld der KI

In diesem Kapitel werden für die Bachelorarbeit wichtige Begriffe aus dem großen Themengebiet der KI erläutert. Dies stellt einerseits eine Basis für ein Grundverständnis dar und soll andererseits dazu dienen die Bachelorarbeit, mit der verwendeten Methode, in das sehr weitläufige Gebiet der KI besser einzuordnen.

### KI

Eine einfache universelle Definition von KI gibt es nicht. Das liegt daran, dass der Begriff Intelligenz selbst nur schwer zu definieren ist. Die Fraunhofer-Gesellschaft beschreibt KI in einer Studie als „ein Teilgebiet der Informatik mit dem Ziel, Maschinen zu befähigen, Aufgaben »intelligent« auszuführen“ [8, Seite 8]. Dabei wird versucht die menschliche Intelligenz bzw. die Entscheidungsstrukturen des Menschen mithilfe von Algorithmen und mathematischen Modellen in Computersystemen nachzubilden und diese auf eigene Problemstellungen anzuwenden. Im Zusammenhang mit KI wird zwischen schwacher und starker KI differenziert. Schwache KIs können nur in einem Bereich ein konkretes Anwendungsproblem bewältigen. D.h. sie sind nicht in der Lage Erlerntes auf andere Bereiche zu übertragen. Typische Anwendungsgebiete sind: Schach/- GO-Computer, Bilderkennung, Spracherkennung, personalisierte Werbung und automatisierte Übersetzung.

Eine starke KI hingegen besitzt intellektuelle Fähigkeiten auf Augenhöhe eines Menschen. Diese beruhen darauf, dass die KI in der Lage ist erlerntes Wissen in andere Bereiche zu übertragen und auf neue Probleme anzuwenden. Jedoch existieren solche starken KIs zurzeit noch nicht. Ob und wann es eine solche allgemeine Intelligenz geben wird, ist in der Wissenschaft umstritten [9]. Der Streitpunkt ist dabei, inwiefern sich typisch menschliche Eigenschaften, wie das Vorhandensein eines Bewusstseins, mit einer KI vereinbaren lassen.

Die Anfänge in der Forschung und Entwicklung der KI lassen sich auf das Jahr 1956 datieren, als am Dartmouth College eine Konferenz namens Dartmouth Conference stattfand. Einen ersten Boom verzeichneten KIs in den 1980er Jahren über die sogenannten Expertensysteme. Die Entscheidungen, welche von den Expertensystemen getroffen werden, basieren auf einer langen Kette von Implikation. Im zweiten KI-Winter sank das öffentliche Interesse an KI und damit auch die Gelder für die Forschung. Durch die Verwendung von maschinellem Lernen, einer weiteren bedeutsamen Untergruppe der KI, erlebt diese in dem 21. Jahrhundert eine Blütezeit (siehe Abbildung 1).



Abbildung 1: Entwicklung der Publikationen im Bereich maschinelles Lernen für Deutschland (DE), Frankreich (FR), Großbritannien (UK), USA (US) und China (CN) sowie die restlichen EU-Staaten und übrige Länder  
Quelle: Frauenhofer…. S.

### maschinelles Lernen

Maschinelles Lernen stellt ein wichtiger Zweig von KI dar. Abbildung 2 veranschaulicht das grundlegende Prinzip des maschinellen Lernens.



Abbildung 2: Grundlegendes Prinzip des maschinellen Lernens; icon quelle

Ein IT-System, bestehend aus Lernalgorithmen und mathematischen Modellen, bekommt als Input einen Datensatz und ist in der Lage in diesem Muster und Zusammenhänge zu erkennen. Das erlernte Wissen wird von dem System genutzt, um selbständig ein Modell zu trainieren, welches eine mathematische Beschreibung der im Datensatz gefundenen Gesetzmäßigkeiten ist. Dieses Modell ist der Output des Systems und kann anschließend, auf für das System unbekannte Daten, angewendet werden, um neue Erkenntnisse zu gewinnen. Zusammenfassend kann man sagen, dass der große Unterschied von maschinellem Lernen zu herkömmlichem Programmieren ist, dass der Mensch dem IT‑System keine direkten Regeln vorgibt, nach denen ein bestimmter Output auszugeben ist, sondern, dass das IT‑System diese selbständig finden muss.

Mit der Technik des maschinellen Lernens ist es möglich riesige Datenmengen effizient zu bearbeiten, was dem Menschen schlicht aufgrund der großen Menge nicht möglich wäre. Maschinelles Lernen lässt sich heutzutage in Bereichen wie der Bilderkennung oder der Textanalyse finden.

Ein viel zitiertes Beispiel ist der Einsatz in Antispamfiltern für E-Mails [10]. Hierfür wird ein Modell mit einem Trainingsdatensatz, bestehend aus vielen E-Mails und dem zugehörigen Label *Spam‑E‑Mail* oder *keine Spam‑E‑Mail*, trainiert. Das trainierte Modell ist dann in der Lage bei einer unbekannten E‑Mail zu erkennen, ob es sich um eine *Spam‑E‑Mail* handelt, oder nicht.

Die klassischen Schritte des maschinellen Lernens sind:

**Datenerfassung:** Überführen der Rohdaten, inklusive der Label, in eine maschinenlesbare Form.

**Datenaufbereitung:** Vorbereitung der Daten (z.B. Normalisieren und Durchmischen der Reihenfolge); Am Ende wird der fertige Datensatz auf einen getrennten Test- und Trainingsdatensatz aufgeteilt. Das Verhältnis in dem zufällig gesplittet wird ist standardmäßig 70 % Trainingsdatensatz zu 30 % Testdatensatz, kann aber auch variiert werden. Es ist wichtig sicherzustellen, dass die Verteilung der Daten zwischen den beiden erzeugten Datensätzen ähnlich ist. Eine Möglichkeit die Ähnlichkeit zu bewerten, ist der Vergleich der Anteile der identischen Label in beiden Datensätzen. Bei dem oben beschriebenen Antispamfilter, stellt man so sicher, dass der Anteil der Spam-E-Mails in beiden Datensätzen ähnlich groß ist. Es wird vermieden, dass das Modell einseitig trainiert wird und dadurch eine spätere Bewertung des Models weniger aussagekräftig werden würde.

**Auswahl des passenden Modells:** Je nach Anforderung, wird aus mehrenden Modellen ein passendes ausgewählt. In Abbildung 3Fehler: Verweis nicht gefunden sind einige Modelle hierarchisch dargestellt. Grundsätzlich wird, wenn die Daten nicht gelabelt sind, von unüberwachtem Lernen gesprochen. Der Antispamfilter ist ein Beispiel für überwachtes Lernen, da wie oben beschrieben, die Daten gelabelt sind. Ein überwachtes Lernen Modell wird mit gelabelten Daten trainiert, so dass es zukünftig zu nicht gelabelten Daten ein Label als Output vorhersagen kann. Innerhalb des überwachten Lernens wird zwischen den beiden Aufgabentypen Regression und Klassifizierung differenziert. Das Entscheidende einer Klassifizierung, zu der auch der Antispamfilter gehört, ist, dass die Anzahl möglicher Label endlich ist. Bei einer Regression hingegen weist das Modell dem Datenpunkt einen stetigen Wert zu, d.h. das Label hat unendlich viele Ausprägungen. Die für diese Arbeit wichtigen Modelle sind Kern-Methoden und Neuronale Netze. Bei beiden handelt es sich um überwachtes Lernen in Verbindung mit einer Klassifizierungsaufgabe.



**Abbildung 3:** Übersicht über verschiedene Modelle des maschinellen Lernens. Die für diese Arbeit relevanten Modelle sind in orangener Farbe dargestellt. Quelle (selbst gemacht, Vorlage aus dem Paper)

**Trainieren des Modells:** Beim Modelltraining wird das Modell mit dem Testdatensatz dahingehend optimiert, dass es später möglichst gut mit den unbekannten Daten zurechtkommt. Es kann beim Trainieren des Modells zu den in Abbildung 4 gezeigten Szenarien kommen.



**Abbildung 4:** Unteranpassung *(underfit),* gute Anpassung *(good fit)* und Überanpassung *(overfit)* beim Modelltraining einer Regression. Die blauen Punkte sind die genauen Werte des Datensatzes und die rote Linie gibt die Näherung durch das trainierte Modell an. Jeder Datenpunkt hat die beiden Eigenschaften und . Quelle: Paper

Von einer Überanpassung ist die Rede, wenn das Modell anfängt sich die Beispiele zu merken, anstelle sich die Werte über gefundene Muster und Zusammenhänge selbst zu erschließen. Bei der Vorhersage mit dem Testdatensatz ist das Modell nicht mehr flexibel genug, um eine gute Vorhersage zu treffen [11]. Das bedeutet, dass die Testgenauigkeit sinkt und dadurch die Differenz zwischen Trainings- und Testgenauigkeit signifikant größer wird, was ein Indiz für eine Überanpassung sein kann. Hingegen liegt eine Unteranpassung vor, wenn das Modell unfähig ist, ein Muster in den Trainingsdaten zu erkennen, um daraus einen Trend ableiten zu können. Zu Unteranpassung kommt es, wenn zu wenige Trainingsdaten vorhanden sind, oder die Komplexität der zu erledigenden Aufgabe, die des Modells wesentlich überschreitet. Das Ziel des Trainings ist es, eine Anpassung zu erreichen, die genau zwischen der einer Überanpassung und der einer Unteranpassung liegt.

**Bewertung des Modells:** Zur Bewertung wird das Modell mit gelabelten Daten des Testdatensatzes getestet, welche bei der Datenerfassung von dem gesamten Datensatz abgetrennt wurden. Dies dient der Abschätzung, wie sich das Modell bei einer tatsächlichen Anwendung verhalten könnte. Als Ergebnis erhält man mehrere Maßzahlen, welche die Leistung des Modells beschreiben (siehe 3.1.4).

Mit einer Validierung werden die übergeordneten Parameter zur Steuerung des Models *(Hyperparameter)* überprüft und gegebenenfalls angepasst, sodass das beste Model gefunden werden kann. Zusätzlich kann das Ergebnis der Validierung als Indikator zur Bewertung des Modells herangezogen werden. Die Validierung kann mit einem Datensatz durchgeführt werden, welcher wie der Testdatensatz, dem Trainingsdatensatz vorenthalten wurde (Houldout-Methode). Das Aufteilen auf diese Teil-Datensätze bringt oftmals das Problem mit sich, dass der Trainingsdatensatz zu klein wird, was in einer Unteranpassung resultieren würde. Um das zu vermeiden, kann die Validierung auch mit einem wechselnden Teil des Trainingsdatensatzes erfolgen, dies wird als Kreuzvalidierung bezeichnet (siehe 3.3.1).

**Vorhersage:**  Das Modell wird an unbekannten Daten angewandt.

Im Folgenden wird die Funktionsweise der beiden Modelle Kern-Methoden und neuronale Netze näher erläutert.

#### Kern-Methoden

Mithilfe von Kern‑Methoden *(engl. kernel‑methods)* lassen sich nicht linear klassifizierbare Daten klassifizieren. Für das Verständnis von Kern‑Methoden ist es sinnvoll erst die lineare Klassifizierung zu betrachten.

Bei einer **linearen Klassifizierung** ist das Ziel eine Hyperebene zu finden, welche die Datenpunkte eines Datensatzes in zwei Klassen einteilt (binäre Klassifizierung). Im zweidimensionalen Vektorraum ist diese Hyperebene eine Gerade (siehe Abbildung 5). Allgemein können Hyperebenen mit der Funktion beschrieben werden. Später kann anhand des Vorzeichens der Funktion bei Einsetzung eines zu klassifizierenden Datenpunktes , die Klasse bestimmt werden. Um die optimale Hyperebene zu finden, muss der Rand (Margin) um die Klassengrenzen möglichst breit werden *(engl. large‑margin‑classification)* [12] , bzw. , was dem Margin entspricht, maximiert werden. Mithilfe des Sattelpunktes der Lagrange-Funktion [13] lassen sich die beiden Stützvektoren definieren, die sich bei einer Margin von befinden und somit die Hyperebene umgeben (siehe Abbildung 5).



**Abbildung 5:** Lineare Klassifizierung eines Datensatzes mit zwei Eigenschaften pro Datenpunkt. Variablen: (Bias und Richtungsvektor der Hyperebene )

Die *large‑margin‑classification* erreicht ihre Grenzen, wenn bereits wenige Ausreißer im Datensatz vorhanden sind. Der Margin würde sehr klein werden und es würde sich eine große Ungenauigkeit für die Klassifizierung ergeben. Die Einführung der Schlupfvariable , welche einerseits Fehleinordnungen erlaubt, diese aber andererseits auch bestraft, stellt eine Möglichkeit dar, diesen potenziellen Fehler zu vermindern. ist Bestandteil der *soft‑margin‑classification* (engl.), einem flexibleren Konzept der Klassifizierung.

Das Modell kann auch in höhere Dimensionen angewandt werden. Die Größe der Dimension des verwendeten Vektorraums wird festgelegt durch die Anzahl der Eigenschaften, die für jeden Datenpunkt gegeben sind (siehe Achsenbeschriftung in Abbildung 5). Die Dimension der Hyperebene ist dabei immer um eins geringer als die Dimension des Modells. Im dreidimensionalen Vektorraum hat die Hyperebene so die Gestalt einer Ebene, oder allgemein ausgedrückt: In der ‑ten Dimension hat die Hyperebene genau Dimensionen.

Oftmals sind die Datenpunkte eines Datensatzes nicht linear trennbar (siehe Abbildung 6). Bei der **nichtlinearen Klassifizierung** kommt der sogenannte Kernel-Trick zum Einsatz. Der Trick besteht darin, die Daten in einen höher dimensionalen Raum zu transformieren, in dem eine lineare Trennung (siehe linearen Klassifizierung ) möglich ist. Die Problematik ist, dass die Berechnung der genauen Datenpunkte in der höheren Dimension sehr rechenintensiv ist [12]. Für das Bestimmen der Hyperebene in der höheren Dimension werden diese genauen Datenpunkte gar nicht benötigt, vielmehr genügt es die Skalarprodukte der Datenpunkte zu kennen. Die Kernel-Funktion ist eine Funktion, welche die jeweiligen Skalarprodukte bereits aus den untransformierten Datenpunkten berechnen kann [14].

Es gibt verschiedene Arten von Kernel-Funktionen, wie z.B. lineare Kernel, polynomiale Kernel, Sigmoid Kernel oder die Gaußsche Radiale Basis-Funktion (RBF).



**Abbildung 6**: Nicht linear trennbare Daten eines Datensatzes mit zwei Eigenschaften pro Datenpunkt.

#### Neuronale Netze

Neuronale Netze, besser *künstliche neuronale Netze (KNN)*, bezeichnen eine Art von Modellen des maschinellen Lernens. KNN wurden ihrem Vorbild, dem menschlichen Gehirn, nachempfunden. Die Lernfähigkeit des menschlichen Gehirns resultiert daraus, dass in dem riesigen Netzwerk aus Neuronen und Synapsen, die Synapsen die Effektivität der Signalübertragung zwischen den Neuronen variieren können [15]. So werden neue synaptische Verbindungen geschaffen und andere getrennt oder vermindert. Ein einzelnes Neuron kann über Dendriten Signale anderer Neuronen empfangen, diese in dem Zellkörper verarbeiten und über die Axone ein resultierendes Signal an andere Neurone weiterleiten. Im Menschen existiert so ein riesiges Netz aus Neuronen und Synapsen.

Das einfachste Modell eines künstlichen Neurons wurde bereits 1943 von Warren McCulloch und Walter Pitts beschrieben [16]. Ihr Neuronenmodell folgt dem Alles-oder-nichts-Prinzip: Nur, wenn die Summe der Eingaben in das Neuron einen festen reellen Schwellenwert überschreitet, feuert das Neuron, d.h. es wird eine 1 weitergegeben. Bleibt der Schwellenwert unterschritten, wird eine 0 ausgegeben. Die Eingabe und Ausgabe sind stets binär. Mit Hilfe eines kleinen Netzes dieser Neuronen lassen sich so bereits einfache logische Verknüpfungen wie UND oder ODER darstellen.

1958 veröffentlichte Frank Rosenblatt das Modell des Perzeptrons [17]. Es beschreibt ein künstliches Neuron, bei dem nicht nur binäre Ein- und Ausgabewerte vorkommen. Außerdem kommt es zu einer Gewichtung der Eingabewerte, wodurch jeweils die Stärke des Einflusses der einzelnen Eingangssignale festgelegt ist. Ein einzelnes Neuron dieses Modells *(**einlagiges Perzeptron)* stellt bereits ein linearer Klassifikator dar. Es ist in Abbildung 7 dargestellt.



**Abbildung 7:** Einlagiges Perzeptron.

Jedem Eingangssignal des Perzeptrons wird ein Gewicht mit zugeordnet. In der Eingangsfunktion wird die gewichtete Summe über die gesamte Eingabe gebildet. Zusätzlich wird auf diese Summe der Bias addiert. Der Bias gibt den Grenzwert an, ab dem das Perzeptron feuern soll.

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

Mithilfe einer Aktivierungsfunktion wird die Ausgabe des Perzeptrons bestimmt:

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

Als Aktivierungsfunktion können verschiedene Funktionen Verwendung finden. Die Sigmoidfunktion findet häufig in modernen KNN Anwendung.

|  |  |  |
| --- | --- | --- |
|  |  | (1.3) |

Bei der Sigmoidfunktion handelt es sich um eine differenzierbare, nicht lineare Funktion. Sie führt, verglichen mit einer einfachen Schwellenwertsfunktion, zu einer weicheren Aktivierung des Neurons [18]. Das Modell des Perzeptrons bildet die Grundlage für KNN.

Mit einem einfachen einlagigen Perzeptron lässt sich noch keine XOR-Verknüpfung (exklusiv-ODER-Verknüpfung) darstellen. Um diese darstellen zu können wird ein dreilagiges Netz an Perzeptren benötigt. Ein solches ist in Abbildung 8 dargestellt.



**Abbildung 8:** Mehrlagiges XOR-Netz. Achtung, hier stellt jeder Kreis ein komplettes Perzeptron dar (wichtiger Unterschied zu Abbildung 7). x und y stehen für die Eingabeoperanden 1 (wahr) und 0 (falsch). Auf den Pfeilen ist je das Gewicht angegeben. Die Neuronen der verdeckten Schicht besitzen zudem einen Schwellenwert, welcher in dem jeweiligen Kreis angegeben ist. Allgemein ist der Schwellenwert nichts anderes als der negative Wert des oben erwähnten Bias. Neuronen der verdeckten Schicht feuern nur, wenn ihre Eingabe größer gleich dem Schwellwert ist. ist die Variable für die gewichtete Summe der Ausgabewerte des Netzes.

Wenn die beiden Eingabewerte identisch sind, ergeben sich für die beiden Fälle:

1. : Kein Perzeptron feuert.
2. :Alle Perzeptren feuern.

Wenn sich die beiden Eingangswerte unterscheiden, ergeben sich für folgende Fälle:

1. : Nur das obere Perzeptron feuert.
2. : Nur das untere Perzeptron feuert.

**Aufbau eines KNN:** Das in Abbildung 8 gezeigte mehrlagige Perzeptron ist ein KNN und besteht aus drei Schichten (eine Eingabeschicht, eine verdeckte Schicht und eine Ausgabeschicht). In der Eingabeschicht befinden sich Perzeptren, die von keinem anderem Perzeptron einen Input erhalten. Sie dienen der Netzeingabe. Die sich an die Eingabeschicht anschließenden verdeckten Schichten eines KNNs (bei dem XOR-Netz ist das eine einzige), entsprechen den bereits oben beschriebenen einlagigen Perzeptren. Allgemein erhalten die Neurone der -ten verdeckten Schicht die Ausgabe der Neurone der -ten verdeckten Schicht als Eingabe. Für jede verdeckte Schicht wird nun mit ( 1 .1) die gewichtete Summe aller verschiedenen Eingaben berechnet und anschließend mit der Aktivierungsfunktion ( 1 .2) die Ausgabe bestimmt. Die Neurone der Ausgabeschicht geben den erhaltenen Wert aus.

Bei KNNs wird allgemein zwischen zwei verschiedenen Klassen unterschieden: Es gibt rekurrente neuronale Netze *(engl. recurrent neural network, RNN)* und vorwärtsgerichtete neuronale Netze *(engl. feedforward neuronal network, FNN).* Sie unterscheiden sich in der Verknüpfung der einzelnen Neurone. Bei dem RNN fließen die Signale stets nur in eine Richtung, während in einem FNN der Signalfluss in Zyklen erfolgt. D.h. es sind auch Verbindungen zu Neuronen derselben oder einer vorangegangenen Schicht möglich. Die heutzutage häufig vorkommende Bezeichnung tiefes Lernen *(engl. deep learning),* beschreibt ein KNN, das über viele verdeckte Schichten verfügt. Je größer die Anzahl der verdeckten Schichten und der Neuronen ist, desto besser können Muster erkennt werden, oder Objekte klassifiziert werden. So basiert z.B. der zu Beginn der Einleitung beschriebene GO-Computer auf tiefen neuronalen Netzen.

**Trainieren von KNN:** Generell ist es wichtig, eine genügend große Menge an Trainingsdaten zu haben, um ein Modell für das tiefe Lernen erfolgreich zu trainieren [19]. Das Training kann auch als Optimierungsproblem aufgefasst werden, bei dem die Gewichte der einzelnen Neurone so verändert werden sollen, dass der Fehler der Klassifizierung möglichst gering wird. Das bedeutet, dass ähnliche Dateneingangsströme von dem Netz aus Neuronen ähnlich behandelt werden sollen. Es erfolgt häufig mit dem sogenannten Gradientenabstiegsverfahren *(engl. backpropagation)*. In jedem Trainingsschritt wird, wie oben beschrieben, ein Dateneingangsstrom von der Eingabeschicht zur Ausgabeschicht verarbeitet und weitergeleitet. Anschließend wird, mit einer Fehlerfunktion die Abweichung der Ausgabe der Ausgabeschicht von dem erwarteten Ergebnis, berechnet. Die Abweichung wird, begonnen mit der Ausgabeschicht, Neuron für Neuron rückgerechnet und dabei jeweils die Gewichte entsprechend der Abweichung angepasst. Dies wird für jeden neuen Dateneingangsstrom wiederholt, bis ein lokales Minimum des Fehlers gefunden wurde. Da vor dem ersten Trainingsschritt noch keine Abweichung berechnet werden kann, werden die ersten Gewichte zufällig gesetzt. Für die Gewichtsänderung selbst ist die Lernrate eine wichtige Größe. Sie gibt an, wie stark die Gewichte jeweils angepasst werden, was einen großen Einfluss auf die Dauer des Lernprozesses hat. Ist sie zu groß gewählt kommt es zu großen Sprüngen und ein Minimum wird möglicherweise nicht gefunden. Wenn sie zu klein ist, kann der Lernprozess viel Zeit beanspruchen.

## Wirkstoffsuche im Zeitalter von Big Data

### klassischer Weg

Das Finden und Entwickeln neuer Wirkstoffe ist in der heutigen Zeit ein Prozess, welcher Jahre dauert und eine große Menge Geld benötigt. Eine Publikation aus dem Jahr 2020 beziffert die durchschnittlichen Kosten für die Entwicklung und Forschung eines neuen Wirkstoffes auf 985 Millionen US-$ [20]. Wenn ein Zielmolekül (meist ein Protein) im Körper identifiziert wurde, geht es darum eine passende Leitstruktur zu finden, welche mit dem Zielmolekül interagieren kann. Dafür kommt seit Mitte der 90er Jahre das Hochdurchsatz-Screening zum Einsatz, bei dem große Molekülbibliotheken mittels automatisierten in vitro-Testsystemen durchmustert werden. Um den fertigen Wirkstoff zu erhalten muss die gefundene Leitstruktur hinsichtlich ihrer pharmakologischen und physikochemischen Eigenschaften optimiert werden. Bei dem fertigen Wirkstoff selbst handelt es sich meistens um eine niedermolekulare Verbindung *(engl. small molecule)*, d.h. eine chemische Verbindung mit einem geringen Molekulargewicht. Die Verbindungen greifen an Zielmolekülen wie Enzymen, Rezeptoren oder Ionenkanälen an und erzielen so eine Wirkung auf den Organismus.

### virtuelles Screening

Auf dem Weg zum fertigen Wirkstoff stellt das Hochdurchsatz-Screening einen zeitaufwendigen und zeitintensiven Prozess dar [21]. Angesichts der riesigen Größe der Molekülbibliotheken, ist es von Interesse der Pharmabranche, die für eine im Hochdurchsatz‑Screening getestete Substanz anfallenden Kosten von 2 bis 5 US-$, zu minimieren. Daher findet das virtuelle Screening, basierend auf computergestützten Techniken, immer mehr an Bedeutung. Dockingprogramme können die Moleküle der Datenbank als 3D-Modelle in die Bindetasche des Proteins legen und mithilfe einer Scoring-Funktion deren zu erwartende Bindungsaffinität bestimmen. So wird die Zahl der möglichen Wirkstoffkandidaten erheblich reduziert und kann mit kostenintensiveren biologischen Tests weiter reduziert werden.

#### Datensätze als Grundlage

Die Grundlage des virtuellen Screenings sind riesige Datensätze mit Informationen zu der Interaktion von kleinen chemischen Verbindungen mit Proteinen *(engl. compound-protein interaction; CPI)*. Solche Informationen sind bereits in diversen Moleküldatenbänken, wie ChEMBL, DrugBank oder PDBbind gesammelt [22] [23] [24]. Diese Daten haben ihren Ursprung in der riesigen Anzahl an wissenschaftlichen Arbeiten der letzten Jahre. 2018 wurden auf der gesamten Welt insgesamt ca. 2,5 Millionen neue wissenschaftliche Arbeiten publiziert [25]. Angesichts dieser riesigen und immer größer werdenden Zahl ist es mit einem sehr großen Aufwand verbunden, die aktuellen Daten aus den wissenschaftlichen Arbeiten in die Datenbänke zu integrieren.

#### Einsatz von Text Mining und NLP

Der Begriff Text Mining beschreibt die Menge an Techniken zum automatischen Extrahieren von neuen, zuvor unbekannten Informationen aus unstrukturierten Texten [26]. Diese Techniken verwenden häufig auf maschinellem Lernen basierende Algorithmen. In diesem Zusammenhang fällt auch der Begriff der Computerlinguistik *(engl. natural language processing; NLP)*. Er bezeichnet die Verarbeitung der natürlichen Sprache mit einem Computer. Bei NLP wird die Sprache zuerst in kleine sprachlich relevante Einheiten (Token) zerlegt, bevor mit Analysen der Morphologie, Syntax und Semantik die Sprache für die Maschine zunehmend erfassbarer wird [26].

**Die Kombination von Text Mining mit NLP stellt eine gute Möglichkeit dar, die für das virtuelle Screening benötigten Daten der CPI direkt aus wissenschaftlichen Arbeiten zu extrahieren.**

In dieser Arbeit werden zwei auf maschinellem Lernen basierende Methoden zum Text Mining von CPI verglichen. In 1.3 befindet sich ein Überblick über die Entwicklung von CPI Text Mining in der Wissenschaft.

## CPI Text Mining

Im CPI Text Mining geht es, wie in 1.2 beschrieben, um die Extraktion der Informationen zu einer möglichen Interaktion zwischen einer kleinen chemischen Verbindung und einem Protein aus wissenschaftlichen Arbeiten. Die interessante Information ist dabei, ob in dem Satz eine funktionale Beziehung des CPI-Paars beschrieben wird oder nicht (siehe Abbildung 9).



**Abbildung 9:** Beispielsätze mit CPI-Paaren aus wissenschaftlichen Arbeiten. Die vorhandenen Proteine sind mit blau und die kleinen chemischen Verbindungen mit rot hinterlegt. **(1)** Beispielsatz: Hier besteht eine funktionale Beziehung (d.h. Interaktion) zwischen dem Protein „insulin“ und der Verbindung „dithiothritol“. **(2)** Beispielsatz mit zwei CPI-Paaren. In beiden besteht keine funktionale Beziehung zwischen dem Protein „BDNF“ und den beiden Verbindungen „MHPG“ und „HVA“.

Bei dem CPI Text Mining handelt es sich um eine konkrete Anwendung von allgemeinem Beziehungs Text Mining *(engl. relation extraction; RE).* Weitere mögliche extrahierte Beziehungen sind z.B. Protein-Protein Interaktionen *(engl. protein-protein interactions)*, Gen-Krankheit Zusammenhänge *(engl. gene-disease associations)* und chemische Verbindungen‑Gen Interaktionen *(engl. compound-gene interaction).* Allgemein werden die einzelnen Objekte (Protein, Gen, Krankheit und chemische Verbindungen) auch als Entitäten bezeichnet. Wenn in einer wissenschaftlichen Arbeit ein Modell zu einer konkreten RE Aufgabe publiziert wird, lässt sich dieses mit kleinen Adaptionen auf weitere Beziehungen anwenden. Bei den auf maschinelles Lernen basierenden Modellen muss auf jeden Fall ein neuer passender Trainingsdatensatz für das Modell vorhanden sein.

### Historie von RE Text Mining mit Blick auf CPI Text Mining

Zum RE aus der Literatur gibt es grundlegend verschiedene Ansätze. Im Folgenden wird ein Überblick über verschiedene Publikationen für RE geliefert.

Die Basis für die ersten Arbeiten ist die Kookkurrenz, bei der sobald zwei Entitäten gemeinsam in einem Abschnitt des Textes (z.B. ein Satz) vorkommen, eine Interaktion zwischen diesen angenommen wird [27]. Die Präzision kann durch die zusätzliche Anwesenheit eines Interaktionswortes zwischen dem Entitäten-Paar [28], oder dem Abgleich mit lexikalisch‑syntaktischen Mustern [29] verbessert werden.

Ein weiterer Ansatz ist, unter Verwendung von NLP die Entitäten‑Paare anhand von selbst definierten Regeln in den Sätzen zu identifizieren. Diese Regeln werden von typischen Mustern der Entitäten-Paare in der Literatur abgeleitet. Die Muster können hierfür entweder manuell [30] oder automatisiert [31] gefunden werden. Mit den musterbasierten Ansätzen zum RE Text Mining, lassen sich relativ gut einfache Interaktionen in Sätzen erfassen. Bei komplexeren Sätzen, welche in der Wissenschaft üblich sind, gerät dieser Ansatz schnell an seine Grenzen [32].

Aus diesem Grund werden Klassifikatoren, die an gelabelten Daten trainierte Modelle des überwachten maschinellen Lernens verwenden, immer interessanter. In den ersten Arbeiten mit diesem Ansatz werden z.B. Bayes’sche Netze [33] oder maximale‑Entropie‑basierte‑Methoden [34] verwendet. Außerdem wird der Einsatz von Kern-Methoden-basierte Modellen immer beliebter. *Tikk et al.* vergleichen 13 publizierte Kern‑Methoden zur Extraktion von Protein-Protein Interaktionen [35]. Zwei dieser Methoden, welche je zu guten Ergebnissen geführt hatten, werden in der Arbeit von *Döring K, Qaseem A, Becer M, Li J, Mishra P, Gao M, et al.* zur Extraktion von CPI‑Paaren aus der Literatur verwendet [36] (siehe *Automated recognition of functional compound-protein relationships in literature (2020)*). Erst seit den letzten Jahren werden als Klassifikatoren verschiedene Arten von Modellen des tiefen Lernens verwendet, wie z.B. RNN [37]. Sehr richtungsweisend für NLP ist die Entwicklung eines vortrainierbaren allgemeinen Sprachmodels wie BERT [38] oder ELMO [39], welche unter anderem zu Text Mining verwendet werden können. Das erfolgreiche Spezialisieren von BERT auf biomedizinische Texte gelingt 2019 [40] (siehe *BioBERT: a pre-trained biomedical language representation model for biomedical text mining (2019)*).

Im Folgenden werden die beiden für diese Bachelorarbeit wichtigen wissenschaftlichen Arbeiten [36], [40] näher beschrieben.

#### Automated recognition of functional compound-protein relationships in literature (2020)

In dieser Arbeit von *Döring K, Qaseem A, Becer M, Li J, Mishra P, Gao M, et al.* aus dem Jahr 2020 [36] wurden die beiden Kern-Methoden *all-paths graph* und *shallow linguistic* mittels eines eigen erstellten Datensatzes kreuzvalidiert und anschließend zum CPI Text Mining aus den Titeln und Abstracts aller vor Juli 2019 veröffentlichen PubMed Artikel verwendet. Die Hauptidee dieser Kern‑Methoden ist es, die Ähnlichkeit der beiden Entitäten anhand der Ähnlichkeiten in der Unterstruktur des Satzes zu bestimmen. Bei der *all‑paths graph* Kern-Methode wird ein Abhängigkeits-Graph der Satzstruktur verwendet, um die Ähnlichkeiten mit Hilfe von gewichteten Pfaden zu bestimmen. Die *shallow linguistic* Kern-Methode setzt sich als Summe aus zwei Kernen zusammen, welche nur einen kleinen Bereich an Wörtern um die Entitäten berücksichtigen. Die resultierende Ähnlichkeit des Entitäten-Paars ergibt sich als Skalarprodukt der Ergebnisvektoren beider Kerne. Die Ergebnisse der Kreuzvalidierung sind in Tabelle 1 dargestellt. Die Kombination der beiden Kerne führt zu einem leicht besseren Ergebnis.

**Tabelle 1:** Ergebnisse der Kreuzvalidierung [36]. Alle Angaben sind in Prozent. Es sind nur die Ergebnisse mit den jeweils besten Parametern angegeben. Für eine Beschreibung der einzelnen Werte siehe 3.1.4)

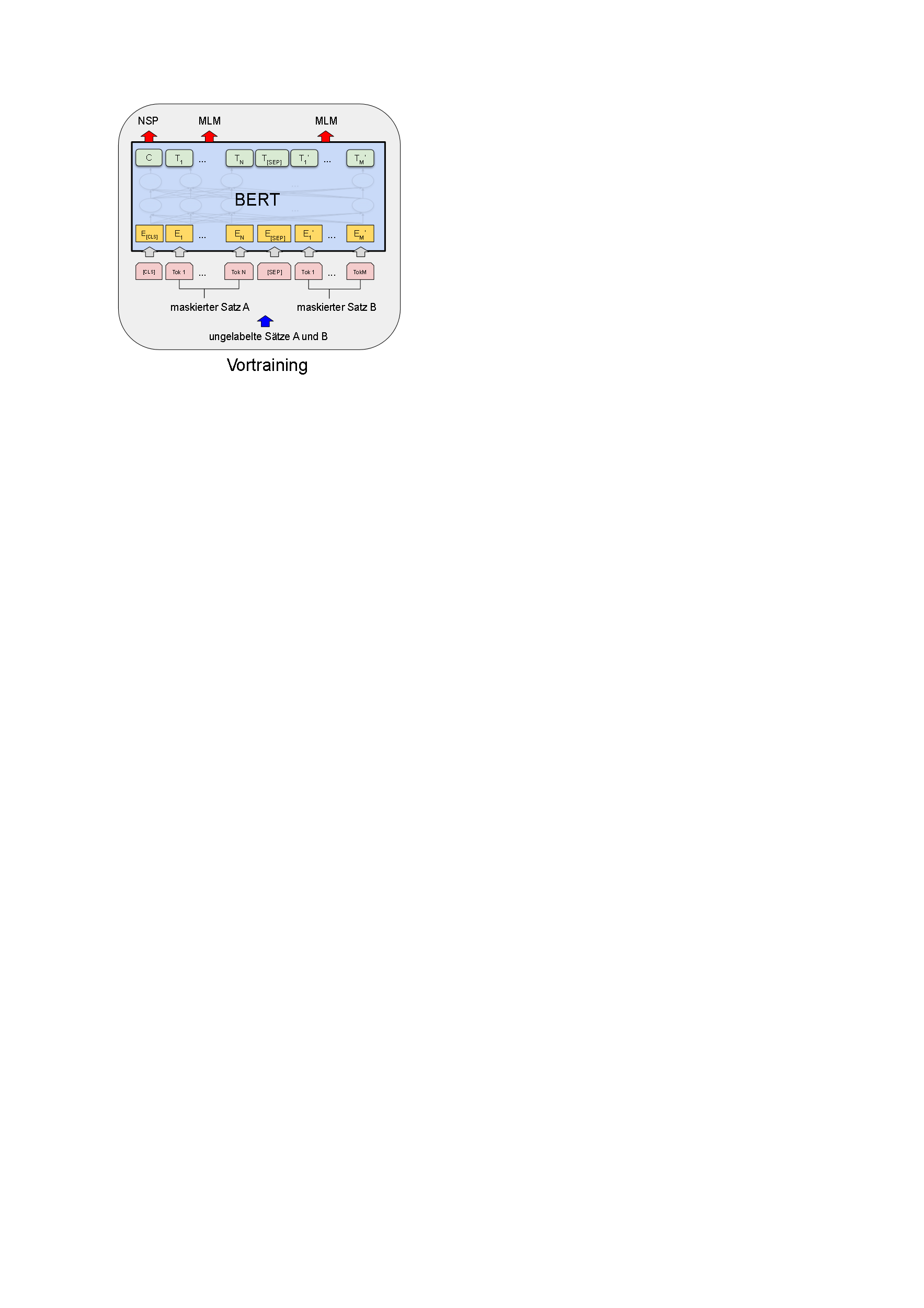
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Kern-Methode** | **Sensitivität** | **Spezifität** | **Genauigkeit** | **F-Maß** | **AUC** |
| *all-paths graph* | 81,7 | 71,8 | **76,6** | 79,0 | 84,6 |
| *shallow linguistic* | 79,5 | 70,2 | **75,0** | 77,2 | 82,5 |
| kombiniert | 68,5 | 81,6 | **80,5** | 74,0 | - |

Der verwendete Datensatz wurde aus den Abstracts der 40000 ersten im Jahr 2009 veröffentlichten PubMed Artikel gebildet und anschließend von Hand mit den Labeln „True“ für eine bestehende funktionale Interaktion oder „False“ für keine funktionale Interaktion versehen. **Das Klassifizieren in binäre Label ist, verglichen mit bereits existierenden CPI-Datensätzen, einzigartig.** So gibt es z.B. in dem häufig verwendeten BioCreativ VI Datensatz [41] zehn verschiedene Label, welche die Art der CPI-Interaktion näher beschreiben. Der Fokus von BioCreativ VI liegt daher auf der genaueren Klassifizierung der gefundenen funktionalen Interaktionen und nicht auf dem Erkennen, ob eine funktionale Interaktionen überhaupt vorliegt.

#### BioBERT: a pre-trained biomedical language representation model for biomedical text mining (2019)

BioBERT ist eine speziell trainierte Version von BERT für NLP von biomedizinischen Texten. Daher entsprechen sich die Aufbauten der beiden Modelle. In BERT werden Transformer ausgiebig genutzt, was, verglichen mit den in ELMO verwendeten *long short-term memories* (bestimmte Art von RNN), zu einer besseren Effizienz führt. Reine Transformer bestehen aus einem Kodierer, zum Lesen des Inputtextes und einem Dekodierer für das Bearbeiten der bestimmten NLP‑Aufgabe. Das Training von Transformern, ist auf die beiden Prozesse Vortraining *(engl. pre‑training)* und Feintuning *(engl. finetuning)* aufgeteilt. Für das Vortraining von BERT (siehe Abbildung 10) werden zwei Trainingsverfahren gleichzeitig durchgeführt:

1. Bei **MLM *(masked language modeling)*** maskiert das Modell zufällig eine Zeichenfolge innerhalb eines als Input erhaltenen Satzes und lernt diese vorherzusagen.
2. Bei **NSP *(next sentence prediction)*** bekommt das Modell zwei Sätze als Input und soll lernen zu prognostizieren, ob der zweite Satz auf den ersten folgt.



**Abbildung 10:** Schemata zum Vortraining von BERT. Die beiden Sätze A und B in die Token aufgeteilt und mache Token maskiert. Die Token werden anschließend mittels Worteinbettung in eine Vektorform überführt und diese dem Kodierer für NSP und MLM zugeführt. Als Output gibt das neuronale Netz die binäre Variable als Ergebnis des NSP und als Vorhersage des maskierten Tokens an.

Für das Vortraining werden riesige Datenmengen ungelabelter Sätze benötigt. Bei BERT wurde hierfür ein Datensatz aus über 11000 Büchern (800 Millionen Wörter), sowie das englische Wikipedia (2,5 Milliarden Wörter) verwendet. Bei BioBERT wurden zusätzlich, je nach Ausführung, noch die PubMed Abtracts (4,5 Milliarden Wörter) oder die gesamten Artikel der PMC‑Datenbank (13,5 Milliarden Wörter) verwendet. Als Ergebnis werden in beiden Arbeiten je die vortrainierten Sprachmodelle, welche gelernt haben, den Kontext eines Wortes in seiner Umgebung zu verstehen, veröffentlicht.

Beim Feintuning wird das Sprachmodell mit einem gelabelten Datensatz auf eine konkrete Aufgabe, wie z.B. dem CPI­ Text Mining, trainiert. Dies wird im Kapitel Material und Methoden verwendet, um das BioBERT-Modell auf das CPI Text Mining vorzubereiten.

# Zielsetzung der Arbeit

Diese Bachelorarbeit soll, wie in der „Conclusion“ [36] beschrieben, als Vergleich von Kern-Methoden mit auf tiefes Lernen basierende Ansätze dienen. Das verwendete Modell des tiefen Lernens ist dabei das vortrainierte BioBERT-Modell [40]. Um eine Vergleichbarkeit zu erreichen, wird der identische Datensatz verwendet. Es ergeben sich folgende Zwischenziele:

1) Überführen des Datensatzes in die notwendige Form

2) Evaluieren des BioBERT-Modells über eine 10-fache Kreuzvalidierung mit dem fertigen Datensatz

3) Bestimmen des Einflusses der Größe des verwendeten Trainingsdatensatzes auf die Klassifikation

# Material und Methoden

In Kapitel 3.1 werden die Ausgangsbedingungen des praktischen Teils der Bachelorarbeit beschrieben, bevor in 3.2, 3.3 und 3.4 die Vorgehensweise beim Erreichen der Zwischenziele dargestellt ist.

## Grundlagen

### Systemeigenschaften

Betriebssystem: Ubuntu 18.04.5 LTS (Bionic Beaver)[[1]](#footnote-2)

Prozessor: Intel® Core™ i5-6600K CPU @ 4.20GHz

Grafikkarte: NVIDIA® GeForce® GTX 1060 6GB

Programmiersprache: Python 3.6.9

### Verwendete Software

Bei dem verwendeten Modell handelt es sich um BioBERT-Base v1.1 (+ PubMed 1M). Dieses stammt, zusammen mit der Software für das Feintuning, von dem offiziellen GitHub BioBERT Repository[[2]](#footnote-3). Für diese Arbeit wurden die Skripte an wenigen Stellen ergänzt, bzw. neue Skripte hinzugefügt. Zudem wurde die Ordnerstruktur verändert. Jede Änderung wird an der entsprechenden Stelle in dieser Arbeit angegeben. Die Software, wie sie Letzen Endes zum Einsatz kam, ist auf GitHub in einem neuen Repository verfügbar.[[3]](#footnote-4)

Als Framework der Arbeit dient die TensorFlow Bibliothek, welche von der Google Brain Abteilung stammt, aber als Open Source Projekt weiterentwickelt wird. TensorFlow ist in Python und C++ geschrieben. Die Verwendung von BioBERT mit Pytorch, einer weiteren bekannten Open Source Bibliothek für maschinelles Lernen, ist ebenfalls möglich[[4]](#footnote-5).

Grundsätzlich wird für die Arbeit mit BioBert eine virtuelle Umgebung verwendet. Das bringt den Vorteil mit sich, einen Überblick über die installierten Python-Pakete zu bewahren und diese besser organisieren zu können. Alle benötigten Pakte, mit den genauen in dieser Arbeit verwendeten Versionen, können der Datei requirements.txt entnommen werden und nach dem Klonen des GitHub Repositorys wie folgt installiert werden:

$ git clone https://github.com/mmmddd98/ba\_git

$ cd ba\_git; pip install -r requirements.txt

### Verwendeter Datensatz

Der, dieser Arbeit zu Grunde liegende Datensatz stammt aus der Publikation von *Döring K, Qaseem A, Becer M, Li J, Mishra P, Gao M, et al.* [36]. Die Struktur der XML‑Datei lässt sich Abbildung 11 entnehmen.



**Abbildung 11:** Ausschnitt von CPI‑DS.xml. „OrigId“ steht für die PubMed-ID, „id“ ist eine eigene Nummerierung über die Artikel. Als „entity“ sind die in dem Satz enthaltenen Entitäten (Proteine und chem. Verbindungen) bezeichnet. Zusätzlich kann dort über „charOffset“ die Position im Satz abgelesen werden. In der „pair“‑Zeile ist bei „interaction“ gezeigt, ob eine funktionale Interaktion des Entitäten-Paars besteht („True“) oder nicht („False).

Informationen zu der Größe des Datensatzes sind in Tabelle 2 dargestellt.

**Tabelle 2:** Inhalt des CPI-DS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Datensatz** | **Anz. Artikel** | **Anz. Sätze** | **Anz. funk. CPIs** | **Anz. keine funk. CPI** | **Ges. Anz. CPI** |
| CPI‑DS | 1808 | 2613 | 2931 | 2631 | 5562 |

### Parameter zur Bewertung von Modellen

Wenn ein maschinelles Lernen Modell trainiert wurde, muss es anhand seiner richtigen und falschen Vorhersagen bewertet werden, um es später mit anderen Modellen vergleichen zu können. Grundsätzlich gibt es bei einer binären Klassifikationsaufgabe mit den Klassen *Positiv* und *Negativ*, vier mögliche Ausgänge der Vorhersage der Klasse:

1. Richtig positiv *(engl. true positv; )*: Richtige Vorhersage der Klasse *Positiv*
2. Falsch positiv *(engl. false positv; )*: Falsche Vorhersage der Klasse *Positiv*
3. Richtig negativ *(engl. true negativ; )*: Richtige Vorhersage der Klasse *Negativ*
4. Falsch negativ *(engl. false negativ; )*: Falsche Vorhersage der Klasse *Negativ*

Daraus lassen sich die folgenden Parameter ableiten.

#### Sensitivität

Die Sensitivität *(engl. recall; )* gibt die Wahrscheinlichkeit an, dass ein als *Positiv* zu klassifizierendes Objekt, richtig als *Positiv* klassifiziert wurde. Alle wurden fälschlicherweise als *Negativ* klassifiziert.

|  |  |  |
| --- | --- | --- |
|  |  | (3.4) |

#### Spezifität

Die Spezifität *(engl. specifity; )* gibt die Wahrscheinlichkeit an, dass ein als *Negativ* zu klassifizierendes Objekt, richtig als *Negativ* klassifiziert wurde. Alle wurden fälschlicherweise als *Positiv* klassifiziert.

|  |  |  |
| --- | --- | --- |
|  |  | (3.5) |

#### Genauigkeit

Die Genauigkeit *(engl. precision; )* gibt die Wahrscheinlichkeit an, dass ein als *Positiv* klassifiziertes Objekt auch wirklich *Positiv* () ist. Dieser Parameter ist wichtig, um ein Klassifikator im Bereich des RE zu bewerten. Der Klassifikator wird hier nur danach bewertet, wie sicher er eine richtige positive Vorhersage trifft. Für die Genauigkeit spielt es keine Rolle, wie viele als *Positiv* zu klassifizierende Objekte, *Falsch* als Negativ vorhergesagt wurden ().

|  |  |  |
| --- | --- | --- |
|  |  | (3.6) |

#### F1-Maß

Das F-Maß, ein kombiniertes Maß, ist das harmonische Mittel von Genauigkeit und Sensitivität . Mit einem Wert zwischen 0 und 1 wird beschreiben, wie gut die Leistung des Modells ist.

|  |  |  |
| --- | --- | --- |
|  |  | (3.7) |

#### AUC

Als AUC *(engl. area under the curve)* wird die Fläche unter der sogenannten ROC‑Kurve *(engl. receiver-operating characteristic curve)* angegeben. Die ROC‑Kurve beschreibt den Zusammenhang der Sensitivität und dem Komplement der Spezifität . Das Integral unter dieser Kurve nimmt Werte zwischen 0 (konstant falsche Vorhersagen) und 1 (ausschließlich richtige Vorhersagen) an. Bei einem Wert von 0,5 würde ein Klassifikator zufällig die Klassen vorhersagen.

## Datensatzpräparation



**Abbildung 12:** Übersicht über die gesamte Arbeit mit dem Datensatz und den dafür verwendeten Python‑Skripten. **Schritt 1** entspricht der Datensatzpräparation. In **Schritt 2** wird der Datensatz zu einem Training- und Testdatensatz mit beliebiger Größe gesplittet (siehe 3.4.1). Mit **Schritt 3a und 3b** wird der Datensatz für die Kreuzvalidierung vorbereitet (siehe 3.3.2 ).

Bei der Datensatzpräparation geht es darum, den Datensatz CPI‑DS.xml, wie er in 3.1.3 beschrieben ist (XML‑Datei), in das für die Verwendung am BioBERT‑Modell notwenige TSV‑Format zu konvertieren. Das wird mit dem Python‑Skript xml\_to\_bert.py durchgeführt. Der fertig konvertierte Datensatz CPI‑DS\_full\_cut.tsv besteht aus drei Spalten. Ein Ausschnitt daraus ist in Abbildung 13 zu sehen. Die wichtigste Änderung ist, dass, unter Verwendung des „charOffset“ (siehe Abbildung 11), die konkreten Namen der Entitäten mit @PROTEIN$ bzw. @COMPOUND$ ersetzt sind. Außerdem wird jeder Satz mit einer ID verknüpft, die für das spätere Splitten des Datensatzes (siehe 2 und 3 in Abbildung 12) benötigt wird.

Das Skript kann in dem Ordner DS\_preparation gefunden werden. Um es zu verwenden, muss in der letzten Zeile des Skriptes in die Funktion main()der zu konvertierende Dateiname CPI‑DS.xml eingefügt werden und das Skript mit folgendem Befehl gestartet werden:

$ cd DS\_preparation

$ python3 xml\_to\_bert.py

Es muss bestätigt werden, dass eine gecuttete Version des Datensatzes erzeugt werden soll. Dies ist notwendig, um Probleme bei dem Ersetzen der Entitäten mit ihren Platzhaltern zu verhindern. Beim Cutten werden 13 Sätze mit CPI‑Paaren aussortiert, bei denen sich das Protein mit der chem. Verbindung überschneitet. Die aussortierten Sätze können durch Bestätigen der Frage "aussortierte Sätze printen?" in dem Terminal ausgegeben werden.



**Abbildung 13:** Ausschnitt aus dem konvertierten Datensatz CPI‑DS\_full\_cut.tsv. Jede Zeile beinhaltet ein CPI‑Paar. Es gibt pro CPI‑Paar drei Einträge: **Eintrag 1** enthält den Satz aus dem Artikel mit dem CPI‑Paar. **Eintrag 2** ist das Label des CPI‑Paars („1“: funktionale Interaktion, „0“: keine funktionale Interaktion). **Eintrag 3** enthält eine artikel- und satzspezifische ID. Wenn ein Satz mehrere verschiedene CPI­‑Paare enthält, erhalten alle dieselbe ID.

Das Splitten dieses konvertierten Datensatzes für das Trainieren von BioBERT ist in Fehler: Verweis nicht gefunden und 3.4.1 beschrieben.

## 10-fache Kreuzvalidierung ~~für den Modellvergleich~~

### -fache Kreuzvalidierung

Bei einer -fachen Kreuzvalidierung wird der Datensatz zufällig in Teilmengen der gleichen Größe gesplittet. Ein Modell wird jeweils mit Teilmengen trainiert (Trainingsdatensatz) und anschließend an der übrigen Teilmenge als Testdatensatz geprüft. Die erhaltenen Bewertungen jedes der -Durchgänge werden anschließend gemittelt. Dadurch wird erreicht, dass der Einfluss von leichteren und schweren Beispielen des Datensatzes gemittelt werden und folglich die allgemeine Fähigkeit des Klassifikators, Muster aus einem Datensatz zu generalisieren, ermittelt werden kann [42].

Für das Klassifizieren von CPI ist es wichtig beim Splitten des Datensatzes alle Interaktionen, welche aus demselben Artikel stammen, in dieselbe Teilmenge zu übertragen. Diese Interaktionen werden als nicht unabhängig angesehen und beim Aufteilen dieser auf verschiedene Teilmengen des Datensatzes wäre eine repräsentative Bewertung des Modells nicht sichergestellt [42].

Generell kann die Varianz von Kreuzvalidierungen, durch mehrfaches Durchführen, verringert werden. In dieser Arbeit werden nach dem beschriebenen Ablauf zwei -fache Kreuzvalidierungen durchgeführt.

### Splitten des Datensatzes für 10-fache Kreuzvalidierung

Anhand von Abbildung 14 kann das Vorgehen für das Splitten des Datensatzes nachvollzogen werden. Im ersten Schritt wird der Datensatz CPI‑DS\_full\_cut.tsv mit dem Skript val\_split\_art.py in zehn gleichgroße Teil-Datensätze gesplittet. Davon werden im zweiten Schritt (3b) jeweils neun zum Trainingsdatensatz zusammengefügt und zusammen mit dem zehnten Teil-Datensatz (Testdatensatz) in insgesamt zehn Ordnern gespeichert. Das Skript val\_ds\_create\_art.py erledigt diese Aufgabe.



**Abbildung 14:** Erzeugen der Kreuzvalidierungsdatensätze. Diese Schritte entsprechen 3a und 3b in Abbildung 12. In grau sind die Teile des Ausgangsdatensatzes dargestellt, welche jeweils zum Trainingsdatensatz zusammengefasst werden. Die orangenen Abschnitte stehen für den Testdatensatz. Für die zweite durchgeführte Kreuzvalidierung werden zehn neue Teile des Ausgangsdatensatzes erzeugt.

#### Erstellen der Teil‑Datensätze mit val\_split\_art.py

Das benötigte Skript befindet sich in DS\_preparation/val\_split und wird, nachdem der Name des zu splittenden Datensatzes in der letzten Zeile in die main()-Funktion eingetragen wurde, mit folgendem Befehl aufgerufen:

$ cd DS\_preparation/val\_split

$ python3 val\_split\_art.py

Anschließend frägt das Skript frägt den Benutzer, wie viele 10-fache Kreuzvalidierungesdatensätze benötigt werden und erstellt diese.

In Abbildung 15 ist die main() Funktion des Skriptes dargestellt. Es lässt sich in drei funktionale Teile unterteilen:

1. **Einlesen des Datensatzes:** Eswird die Eingabedatei ausgelesen und in Listen gespeichert. Wichtig ist, dass die artikelspezifische ID (siehe Abbildung 11) erhalten bleibt.
2. **Erstellen und Auswerten des Splittes:** In der Variable random\_zahlen\_part wird eine Liste mit allen ganzen Zahlen bis zu der Länge der zu splittenden Interaktionen angelegt. Diese wird zufällig auf die zehn Teile aufgeteilt und jeder Zahl als Index einer Interaktion zugeordnet. Dabei wird über die artikelspezifische ID sichergestellt, dass Interaktionen aus dem gleichen Artikel in denselben Teil gelangen. Mit der Funktion auswerten()wird der erzeugte Splitt hinsichtlich seiner Homogenität bewertet. Dafür werden die Standardabweichungen der unterschiedlichen Größen der Teildatensätze, sowie das Verhältnis von funktionalen Interaktionen zu nicht funktionalen Interaktionen, herangezogen. Schritt 2 wird so oft wiederholt, bis bestimmte Grenzwerte (siehe Tabelle 3) unterschritten werden und gespeichert werden kann.

**Tabelle 3:** Grenzwerte Splitten für 10-fache Kreuzvalidierung. SD steht für Standardabweichung. Es wurden je aus den zehn erzeugten Teildatensätzen die einzelnen Merkmale bestimmt und die SD berechnet. Um das Verhältnis der Anzahl von funktionalen zu nicht funktionalen Interaktionen eines Teildatensatzes zu bestimmen, werden die beiden Anzahlen bestimmt und der Quotient gebildet. Die SD über alle dieser zehn Quotienten wird mit dem Grenzwert verglichen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Merkmal** | **SD: Anzahl**  **Interaktionen** | **SD: Anzahl**  **Sätze** | **SD: Verhältnis**  **nicht funkt. / funkt.** |
| Grenzwert | 22 | 10 | 0,12 |

1. **Speichern der Teil-Datensätze:** Die zehn Teil-Datensätze werden gespeichert, zusätzlich wird für jeden Teil eineresults.tsv Datei erstellt, in welcher statistische Daten zu dem Teil-Datensatz angegeben sind.



**Abbildung 15:** main()Funktion aus dem Skript val\_split\_art.py. Die Leerzeilen trennen die drei funktionalen Bereiche (Einlesen des Datensatzes, Erstellen und Auswerten des Splittes und das Speichern der Teil-Datensätze). Die vorletzte Zeile ist gekürzt dargestellt.

#### Erstellen der fertigen Datensätze mit val\_ds\_create\_art.py

Um den Datensatz für BioBERT zu verwenden müssen die zehn Teil-Datensätze nach dem Prinzip der 10-fachen Kreuzvalidierung zusammengesetzt werden. Das erfolgt mit dem Ausführen des Skriptes val\_ds\_create\_art.py. Dafür wird der Name des Ordners mit dem Splitt als Kommandozeilenargument hinter das Skript gesetzt. Z.B.:

$ python3 val\_ds\_create\_art.py 10\_cross\_fold\_val1

Der erzeugte Datensatz ist als neuer Ordner in dem oben angegebenen Ordner 10\_cross\_fold\_val1 zu finden.

### Durchführung der 10-fachen Kreuzvalidierung

Um mit dem erzeugten Kreuzvalidierungsdatensatz das Modell zu trainieren und anschließend zu bewerten wird der fertige Datensatz (als Ordner) in den Ordner Biobert/tensorflow/ds/cross\_val verschoben.

#### Starten der Kreuzvalidierung mit run\_val.py

Für die eigentliche Kreuzvalidierung wird das Skript run\_val.py aus dem Ordner Biobert/tensorflow/biobert\_main benötigt. Vor dem Aufrufen des Skriptes muss in der main()Funktion des Skriptes der Name des Ordners mit dem kompletten Validierungsdatensatz eingefügt werden. Dann wird die Kreuzvalidierung wie folgt gestartet:

$ cd Biobert/tensorflow/biobert\_main

$ python3 run\_val.py

In run\_val.py werden mehrere Skripte der ursprünglichen BioBERT Publikation vereint. So ist es möglich die zehn Schritte der Kreuzvalidierung am Stück laufen zu lassen und dies mit einem Skript starten zu können. run\_val.py setzt sich aus den folgenden Teilen zusammen:

1. **Vorverarbeitung** *(engl. Preprocessing)***:** Hier wird ein Ausgabeordner mit dem Namen des Eingabeordners in Biobert/tensorflow/results/cross\_val angelegt. Zudem werden die einzelnen Datensätze in die für BioBERT notwendigen Namen test.tsv und train.tsv umbenannt, sowie die ebenfalls benötigte dev.tsv Datei eingefügt.
2. **Feintuning:** Hier findet das eigentliche Feintuning von BioBERT, mit train.tsv als Trainingsdatensatz, statt. Die verwendete Lernrate beträgt 0.00002.
3. **Bewertung:** Für die Bewertung des trainierten Modells wird betrachtet, wie gut das Modell die CPI-Paare von test.tsv klassifizieren kann. Dies geschieht unter Verwendung des Skriptes re\_eval.py aus der BioBERT Publikation. Es wurde dahingehend verändert, dass die aus der Bewertung erhaltenen Parameter wie F-Maß oder Genauigkeit, in der Datei scores.tsv im Ausgabeordner abgespeichert werden. Außerdem wird auch die benötigte Zeit für das Feintuning in dieser Datei gespeichert.

#### Auswerten Kreuzvalidierung mit val\_create\_results.py

Mit dem Skript val\_create\_results.py werden alle Ergebnisse der zehn Teildatensätze in einer Datei mit dem Namen results.tsv im Ausgabeordner dargestellt und das Ergebnis der Kreuzvalidierung, durch Bildung des Mittelwertes über die zehn Teilergebnisse gebildet. Der Name des Ordners mit den Teilergebnissen wird als Kommandozeilenargument hinter den Skriptnamen gesetzt:

$ cd Biobert/tensorflow/results/cross\_val

$ python3 val\_create\_results.py 10\_cross\_fold\_val1

### Reproduzieren der Klassifizierungsergebnisse der Kern-Methoden

Um eine gute Vergleichbarkeit zwischen der Kreuzvalidierung des auf tiefes Lernen basierenden BioBERT-Modells und den beiden Kern-Methoden der Arbeit von *Döring K, Qaseem A, Becer M, Li J, Mishra P, Gao M, et al.* zuerreichen, werden die 10-fachen-Kreuzvalidierungen der Kernmethoden wiederholt. Die genaue Implementierung ist in einem GitHub Repository veröffentlicht.[[5]](#footnote-6) Nach dem Installieren der notwendigen Pakte, werden die beiden Kreuzvalidierungen, wie in Punkt „How to run the Kernel Pipelines“[[6]](#footnote-7) beschrieben, durchgeführt. Als Datensatz wird derselbe

## Einfluss der Trainingsdatensatzgröße

Um zu untersuchen, wie sich die Leistung des Modells bei einem größerem, bzw. kleineren Trainingsdatensatz verhält, wird das Verhältnis von Trainingsdatensatzgröße zu Testdatensatzgröße variiert (verwendete Verhältnisse: 0,8; 0,7; 0,6; 0,5; 0,4; 0,3; 0,2; 0,1). Für jedes Verhältnis werden zehn verschiedene Datensätze (je ein Trainingsdatensatz und ein Testdatensatz) zufällig erzeugt und mit dem Trainingsdatensatz je ein Modell trainiert. Jedes trainierte Modell wird anhand seiner Vorhersagen am Testdatensatz bewertet. Das Ergebnis eines Verhältnisses ergibt sich als arithmetisches Mittel der Ergebnisse aus den zehn Vorhersagen.

### Splitten des Datensatzes



**Abbildung 16:** Erzeugen der Datensätze für das Untersuchen des Einflusses der Trainingsdatensatzgröße. Die Teile des Trainingsdatensatzes sind in grau und die des Testdatensatzes in orange dargestellt.

Der Ausgangsdatensatz CPI‑DS\_full\_cut.tsv wird mit dem Skript split\_tsv\_art.py aus dem Ordner DS\_preparation/size\_split in verschieden große Trainings- und Testdatensätze gesplittet. Das Splitten erfolgt, wie bei der Kreuzvalidierung, artikelbasiert.

$ cd DS\_preparation/size\_split

$ python3 split\_tsv\_art.py

Der Anwender muss bestätigen, dass Index Spalten benötigt werden und kann selbst, durch Eingabe in das Terminal, den benötigten Anteil des Testdatensatzes am Ausgangsdatensatz, sowie die gesamte Anzahl an, mit diesem Splitt erzeugten Datensätze, festlegen. split\_tsv\_art.py ähnelt in der Funktionsweise und dem Aufbau dem Skript zum Splitten des Datensatzes für die Validierung. Ebenfalls kommt eine auswerten() Funktion zum Einsatz, um die Homogenität zwischen dem erzeugten Test und Trainingsdatensatz zu bewerten. Aufgrund des artikelbasierten Splittens kann es z.B. sein, dass alle Artikel mit vielen beschriebenen Interaktionen im Testdatensatz landen und dadurch das Verhältnis von Anzahl der Interaktionen in Test- und Trainingsdatensatz deutlich von dem gewünschten Split Verhältnis der Artikel abweicht. Ebenso kann es sein, dass durch das Splitten ein Ungleichgewicht in der Verteilung von funktional und nicht funktional gelabelten Interaktionen entsteht. Um das zu verhindern werden so lange neue zufällige Splits erzeugt, bis bestimmte Grenzwerte eingehalten werden (siehe Tabelle 4). Die Ergebnisse dieser Bewertung werden für alle erfolgreichen Splitts in einer Datei mit dem Namen DS\_results.tsv gespeichert.

**Tabelle 4:** Grenzwerte Splitten für Variation der Trainingsdatensatzgröße. Für die ersten beiden Merkmale werden die Anzahlen der Interaktionen bzw. der Sätze des Testdatensatzes ermittelt und deren Anteil an der Gesamtzahl mit der manuellen Eingabe des Gewünschten Anteils (auf Artikel bezogen) relativ verglichen. Für das dritte Merkmal werden die Quotienten für den Test- und Trainingsdatensatz gebildet. Es wird die relative Abweichung dieses Quotienten zu dem des Ausgangsdatensatzes berechnet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Merkmal** | **Verh. Anzahl**  **Interaktionen** | **Verh. Anzahl**  **Sätze** | **Verh. Verhältnis**  **nicht funkt. / funkt.** |
| maximale relative Abweichung | 1% | 1% | 1% |

### Training und Testen des Modells

Im folgenden wird erläutert, wie für das Splitverhältnis 0,5 verfahren wird. Die Verwendung der Skripte ist für die übrigen sieben betrachteten Verhältnisse analog.

Starten der Trainings mit run\_mult.py

Nach dem der Ordner Split\_10\_0.5, welcher die zehn erzeugte Datensätze beinhaltet, nach Biobert/tensorflow/ds/runs verschoben wurde kann das Skript wie folgt gestartet werden:

$ cd Biobert/tensorflow/biobert\_main

$ python3 run\_mult.py

Zuvor wird der Ordner mit den Datensätzen in die main() Funktion des Skriptes in der letzten Zeile eingefügt. Der Aufbau und die Funktionsweise gleicht der für die Validierung verwendetet Funktion (siehe XXX). Wenn das Skript läuft, werden zehn BioBERT Modelle nacheinander am Trainingsdatensatz trainiert und mit dem Testdatensatz getestet. Die Testergebnisse werden für jeden der zehn Datensätze als test\_scores.tsv Datei in den Ausgabeordner Biobert/tensorflow/results/runs ausgegeben.

Auswerten mit result\_create.py

Anschließend werden die Ergebnisse mit dem Skript result\_run\_create.py in eine Datei zusammengetragen und das arithmetisches Mittel, sowie die Standardabweichung berechnet.

$ cd Biobert/tensorflow/results/runs

$ python3 result\_create.py Split\_10\_0.5

Der Name des auszuwertenden Ordners ist als Kommandozeilenargument hinter das Skript zu setzen.

val\_split und size\_split überall mit einem t!!!!

# Ergebnisse

## Kreuzvalidierung für Modellvergleich

2 graphen mit f1 gegen stat länge  keine auswirkung  DS ist okay gewesen

Runtime!!!!

* Gleicher split wie in fr paper zum dir vergleich
* Validationwaren aber auch zum vergleich nutzen
* Veränderung der DS größe um

# Diskussion und Ausblick

https://www.statistik.tu-dortmund.de/fileadmin/user\_upload/Lehrstuehle/Datenanalyse/Wissensentdeckung/Wissensentdeckung-Li-7.pdf

bewerten wie homogen die datensätze sind über:

-standard abw von int als gutes maß um zu bewerten dass DS homogen ist ?

-noch mit verhältnis von verhältnis von 1/0 aus test und train oder bei val nur ds ?

- bewerten des Modell über precision, da FN keine rolle spielen (nicht wichtig wie viele nicht gefunden wurden hauptsaxhe keine falsch) ?

- oder lieber mit f1 ?

was sind denn überhaupt die anforderungen die du an das modell hast?

cv machen mit random nene

randwom artikel genommen

this ratio enspricht auch der reality

comparison in ru time ==> auch machen von

ziel am ende alle benutzen

biocreativ

gpu experiment als output und weitere möglichkeit

output from biobert verstehen für später wichtig

https://www.merkleinc.com/blog/dispelling-myths-deep-learning-vs-machine-learning

dataset kann noch erhöt werden mit semi manuell

am ende alle 3 ziel

outlook biobert auf alle artikel machen

Overfitting underfitting?

Ml paper generell fig 1b performance kernek´l methiden schlechter als dl

[einleitung\einleitung\_paper\ML\_paper\_generell.pdf](../../../../../../../../C:/Users/Manuel%20Dorer/Desktop/BA_win/schreiben/einleitung/einleitung_paper/ML_paper_generell.pdf)

hyperparamter optimierung noch was rausholen

pytoech vs tensorflow (sollte eig nix machen aber vlt überprüfen)

ML andere vs dl <https://machinelearningmastery.com/what-is-deep-learning/>

<https://arxiv.org/pdf/2101.01907.pdf> dl ganz neues paper überblick

cuda

Datensätze nicht identisch wie in Freiburg  paar rausgeworfen

Vlt risiko mit tensorflow kostenpflichtigkeit ansprechen

<https://pubmed.ncbi.nlm.nih.gov/19881490/>

paper dass zeigt das CPI pairs wichtig für discovery and development of safe and effective drugs

# Zusammenfassung

# Literaturverzeichnis

[1] “Google-Software besiegt Go-Genie auch im letzten Match.” https://www.faz.net/aktuell/gesellschaft/menschen/google-computer-alphago-besiegt-go-weltmeister-14125664.html (accessed Feb. 15, 2021).

[2] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: 10.1038/nature16961.

[3] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey, “WeatherBench: A Benchmark Data Set for Data-Driven Weather Forecasting,” *J. Adv. Model. Earth Syst.*, vol. 12, no. 11, 2020, doi: 10.1029/2020MS002203.

[4] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep reinforcement learning framework for autonomous driving,” *arXiv*, pp. 70–76, 2017.

[5] E. V. Polyakov, M. S. Mazhanov, A. Y. Rolich, L. S. Voskov, M. V. Kachalova, and S. V. Polyakov, “Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning,” *Moscow Work. Electron. Netw. Technol. MWENT 2018 - Proc.*, vol. 2018-March, pp. 1–5, 2018, doi: 10.1109/MWENT.2018.8337236.

[6] A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, 2017, doi: 10.1038/nature21056.

[7] “No Title.” https://www.selfmem.eu/public\_relations\_media/magazine/what\_motivates\_us/081840/index.php.de (accessed Mar. 02, 2021).

[8] I. Döbel *et al.*, “Maschinelles Lernen: Kompetenzen, Forschung, Anwendung,” *Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.*, 2018. https://www.bigdata-ai.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Fraunhofer\_Studie\_ML\_201809.pdf.

[9] R. Fjelland, “Why general artificial intelligence will not be realized,” *Humanit. Soc. Sci. Commun.*, vol. 7, no. 1, pp. 1–9, 2020, doi: 10.1057/s41599-020-0494-4.

[10] “How To Design A Spam Filtering System with Machine Learning Algorithm.” https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472 (accessed Feb. 20, 2021).

[11] “Verhindern von Überanpassung und unausgeglichenen Daten durch automatisiertes maschinelles Lernen.” https://docs.microsoft.com/de-de/azure/machine-learning/concept-manage-ml-pitfalls (accessed Feb. 22, 2021).

[12] “Support Vector Machines (SVM); Universität Ulm.” http://www.mathematik.uni-ulm.de/stochastik/lehre/ss07/seminar\_sl/fischer.pdf (accessed Feb. 24, 2021).

[13] “Support Vector Machines (SVM) Ausarbeitung; Universität Ulm.” http://www.mathematik.uni-ulm.de/stochastik/lehre/ss07/seminar\_sl/ausarbeitung\_fischer.pdf (accessed Feb. 24, 2021).

[14] “The Kernel Trick in Support Vector Classification.” https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f (accessed Feb. 24, 2021).

[15] “Das Gehirn,” *MAX-PLANCK-GESELLSCHAFT*. https://www.mpg.de/gehirn (accessed Feb. 26, 2021).

[16] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity (reprinted from bulletin of mathematical biophysics, vol 5, pg 115-133, 1943),” *Bull. Math. Biol.*, vol. 52, no. 1--2, pp. 99–115, 1990, [Online]. Available: http://journals2.scholarsportal.info/pdf/00928240/v52i1-2/99\_alcotiiina.xml.

[17] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.

[18] A. Scherer and A. Scherer, “Das Perzeptron,” *Neuronale Netze*, pp. 65–70, 1997, doi: 10.1007/978-3-322-86830-5\_5.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2014.

[20] O. J. Wouters, M. McKee, and J. Luyten, “Estimated Research and Development Investment Needed to Bring a New Medicine to Market, 2009-2018,” *JAMA - J. Am. Med. Assoc.*, vol. 323, no. 9, pp. 844–853, 2020, doi: 10.1001/jama.2020.1166.

[21] G. Klebe, *Wirkstoffdesign*, 2.Auflage. Spektrum Akademischer Verlag, 2009.

[22] D. Mendez *et al.*, “ChEMBL: Towards direct deposition of bioassay data,” *Nucleic Acids Res.*, vol. 47, no. D1, pp. D930–D940, 2019, doi: 10.1093/nar/gky1075.

[23] D. S. Wishart *et al.*, “DrugBank 5.0: A major update to the DrugBank database for 2018,” *Nucleic Acids Res.*, vol. 46, no. D1, pp. D1074–D1082, 2018, doi: 10.1093/nar/gkx1037.

[24] R. Wang, X. Fang, Y. Lu, C. Y. Yang, and S. Wang, “The PDBbind database: Methodologies and updates,” *J. Med. Chem.*, vol. 48, no. 12, pp. 4111–4119, 2005, doi: 10.1021/jm048957q.

[25] “Scientific and technical journal articles.” https://data.worldbank.org/indicator/IP.JRN.ARTC.SC?year\_low\_desc=true (accessed Mar. 04, 2021).

[26] D. R. Witte and J. Mülle, “Text Mining: Wissensgewinnung aus natürlichsprachigen Dokumenten,” 2005.

[27] T. Jenssen, J. Komorowski, and E. Hovig, “A literature network of human genes for high-throughput analysis of gene,” vol. 28, no. may, pp. 21–28, 2001.

[28] R. Kabiljo, A. B. Clegg, and A. J. Shepherd, “A realistic assessment of methods for extracting gene / protein interactions from free text,” vol. 12, pp. 1–12, 2009, doi: 10.1186/1471-2105-10-233.

[29] F. Rinaldi *et al.*, “Open Access OntoGene in BioCreative II,” vol. 9, no. Suppl 2, pp. 1–11, 2008, doi: 10.1186/gb-2008-9-S2-S13.

[30] J. M. Temkin and M. R. Gilder, “Extraction of protein interaction information from unstructured text using a context-free grammar,” *Bioinformatics*, vol. 19, no. 16, pp. 2046–2053, 2003, doi: 10.1093/bioinformatics/btg279.

[31] Y. Hao, X. Zhu, M. Huang, and M. Li, “Discovering patterns to extract protein-protein interactions from the literature: Part II,” *Bioinformatics*, vol. 21, no. 15, pp. 3294–3300, 2005, doi: 10.1093/bioinformatics/bti493.

[32] I. Segura-Bedmar, P. Martínez, and C. de Pablo-Sánchez, “Using a shallow linguistic kernel for drug-drug interaction extraction,” *J. Biomed. Inform.*, vol. 44, no. 5, pp. 789–804, 2011, doi: 10.1016/j.jbi.2011.04.005.

[33] R. Chowdhary, J. Zhang, and J. S. Liu, “Bayesian inference of protein-protein interactions from biological literature,” *Bioinformatics*, vol. 25, no. 12, pp. 1536–1542, 2009, doi: 10.1093/bioinformatics/btp245.

[34] C. Sun, L. Lin, X. Wang, and Y. Guan, “Using maximum entropy model to extract protein-protein interaction information from biomedical literature,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4681 LNCS, no. August, pp. 730–737, 2007, doi: 10.1007/978-3-540-74171-8\_72.

[35] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser, “A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature,” *PLoS Comput. Biol.*, vol. 6, no. 7, p. 32, 2010, doi: 10.1371/journal.pcbi.1000837.

[36] K. Döring *et al.*, “Automated recognition of functional compound-protein relationships in literature,” *PLoS One*, vol. 15, no. 3, pp. 1–14, 2020, doi: 10.1371/journal.pone.0220925.

[37] D. Zhang and D. Wang, “Relation Classification via Recurrent Neural Network,” 2015, [Online]. Available: http://arxiv.org/abs/1508.01006.

[38] C. Sun *et al.*, “Chemical-protein interaction extraction via Gaussian probability distribution and external biomedical knowledge,” *Bioinformatics*, vol. 36, no. 15, pp. 4323–4330, 2020, doi: 10.1093/bioinformatics/btaa491.

[39] M. E. Peters *et al.*, “Deep contextualized word representations,” *NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, pp. 2227–2237, 2018, doi: 10.18653/v1/n18-1202.

[40] J. Lee *et al.*, “BioBERT: A pre-trained biomedical language representation model for biomedical text mining,” *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020, doi: 10.1093/bioinformatics/btz682.

[41] “ChemProt corpus: BioCreative VI.” https://biocreative.bioinformatics.udel.edu/media/store/files/2017/ChemProt\_Corpus.zip (accessed Mar. 08, 2021).

[42] Kersten Döring, “Processing Information about Biomolecules with Text Mining and Machine Learning Approaches,” Albert-Ludwigs-Universität Freiburg im Breisgau, 2015.

1. Zuerst wurde in einer VirtualBox unter Windows 10 begonnen, dann mittels einer neuen SSD ein Dual‑Boot‑System mit Ubuntu eingerichtet. [↑](#footnote-ref-2)
2. https://github.com/dmis-lab/biobert [↑](#footnote-ref-3)
3. https://github.com/mmmddd98/ba\_git [↑](#footnote-ref-4)
4. https://github.com/dmis-lab/biobert-pytorch [↑](#footnote-ref-5)
5. https://github.com/KerstenDoering/CPI-Pipeline [↑](#footnote-ref-6)
6. https://github.com/KerstenDoering/CPI-Pipeline#how-to-run-the-kernel-pipelines [↑](#footnote-ref-7)