Albert-Ludwigs-Universität Freiburg

Automatisierte Klassifizierung von Wirkstoff-Protein-Wechselwirkungen 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 31.03.2021, 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:	M. Sc. 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

Kurzfassung

In Moleküldatenbanken wie ChEMBL, DrugBank oder PDBbind [1] [2] [3] sind Informationen zu Wirkstoffen und Proteinen aus wissenschaftlichen Publikationen gesammelt und strukturiert dargestellt. Angesichts der riesigen Menge an Publikationen jedes Jahr ist es ein schwieriges Unterfangen diese Datenbanken aktuell zu halten. Ein vielversprechender Ansatz, um die Extraktion funktionaler Wirkstoff-Protein-Wechselwirkungen aus unstrukturierter Literatur zu automatisieren, ist die Verwendung von Text Mining in Verbindung mit Methoden des maschinellen Lernens. Seit den letzten Jahren werden für diese Aufgabe zunehmend auch Modelle des tiefen Lernens verwendet. In dieser Bachelorarbeit wird die Klassifizierungsleistung von BioBERT, einem Modell des tiefen Lernens, mit zwei Kern-Methoden verglichen. Die Modelle sagen voraus, ob die Wechselwirkung einer chemischen Verbindung mit einem Protein (CPI) funktional oder nicht funktional ist. Zur Evaluierung der Modelle werden 10-fache-Kreuzvalidierungen mit einem Datensatz durchgeführt. In dem zweiten Teil der Arbeit wird untersucht, wie die Menge an Trainingsdaten die Leistung des Modells beeinflusst. Dafür wird BioBERT mit unterschiedlichen Trainingsdatensatzgrößen trainiert und anschließend evaluiert. Die Ergebnisse der Kreuzvalidierungen (F-Maß von 0,85) zeigen eine höhere Klassifizierungsleistung von BioBERT als die der Kern-Methoden (F-Maß von 0,79 und 0,78). Bei der Variation der Trainingsdatensatzgröße wurde festgestellt, dass die Klassifizierungsleistung besonders bei einer kleinen Trainingsdatenmenge niedriger ausfällt. Die hier vorgestellten Ergebnisse legen nahe, dass die Forschung an tiefem Lernen für den Einsatz im Text Mining von CPI ein wichtiger Schritt für die Zukunft ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

KI	künstliche Intelligenz
KNN	künstliches neuronales Netz
RNN	rekurrentes neuronales Netz
FNN	vorwärtsgerichtetes neuronales Netz *(engl. feedforward neuronal network)*
CPI	Interaktion von kleiner chemischer Verbindung mit Protein *(engl. compound-protein interactions)*
NLP	Computerlinguistik *(engl. natural language processing)*
RE	Beziehungs-Text Mining *(engl. relation extraction)*
APG	all-paths graph
SL	shallow linguistic

1 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 [4]. 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 [5] [6] [7] [8]. 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 [9], das Optimieren von Molekülsimulationen für die Werkstoffforschung in der Physik [10], oder generell das Extrahieren von Informationen aus Daten. Besonders letzteres gewinnt im Zeitalter von Big Data immer mehr an Relevanz und soll in dieser Bachelorarbeit im Mittelpunkt stehen.

1.1 Begriffe aus dem Feld der künstlichen Intelligenz

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.

1.1.1 Künstliche Intelligenz

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“ [11, S. 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-Computer, 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 [12]. 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).

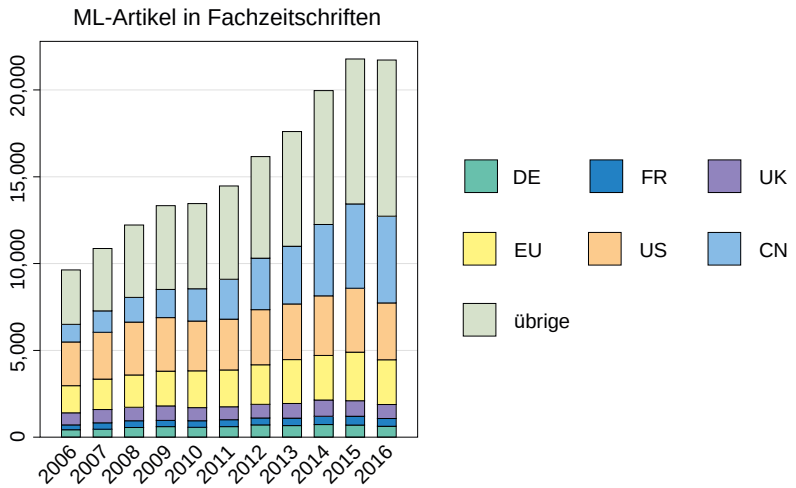

Abbildung 1: Entwicklung der Publikationen im Bereich maschinelles Lernen für Deutschland (DE), Frankreich (FR), Großbritannien (UK), USA (US) und China (CN) sowie die restlichen EU-Staaten und übrige Länder. Quelle: [11, S. 14]

1.1.2 Maschinelles Lernen

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

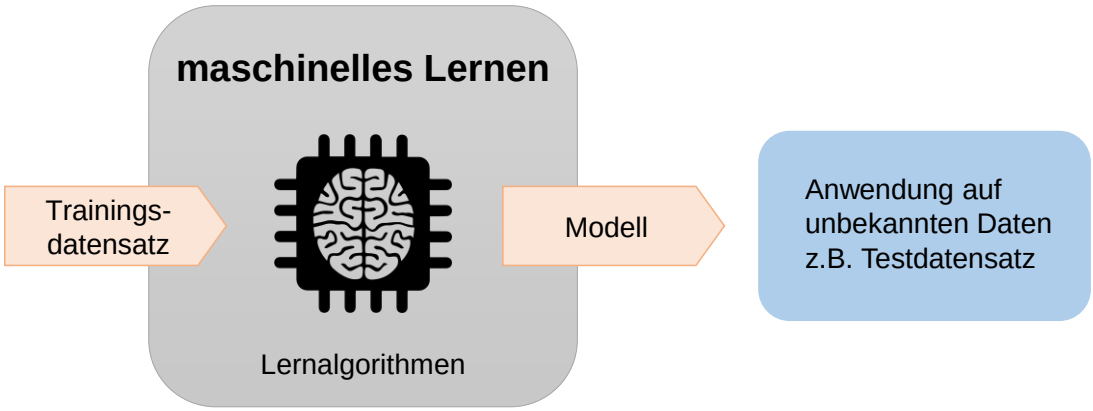

Abbildung 2: Grundlegendes Prinzip des maschinellen Lernens. Quelle: Eigene Darstellung

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 [13]. 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 3 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.

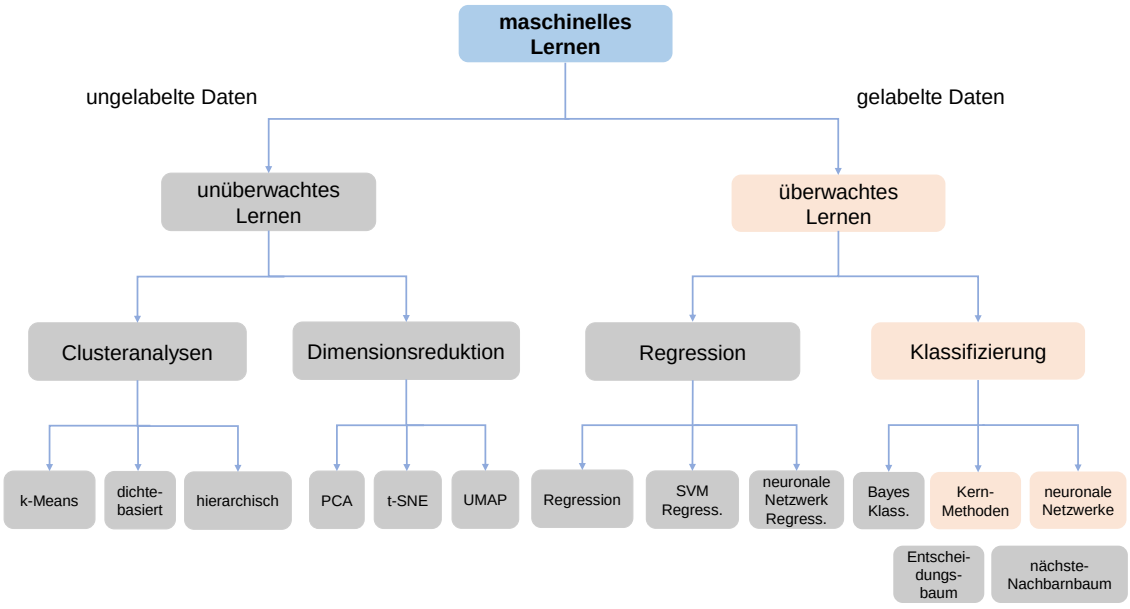

Abbildung 3: Übersicht über verschiedene Modelle des maschinellen Lernens. Die für diese Arbeit relevanten Modelle sind in orangener Farbe dargestellt. Quelle: Eigene Darstellung nach [14, S. 2]

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.

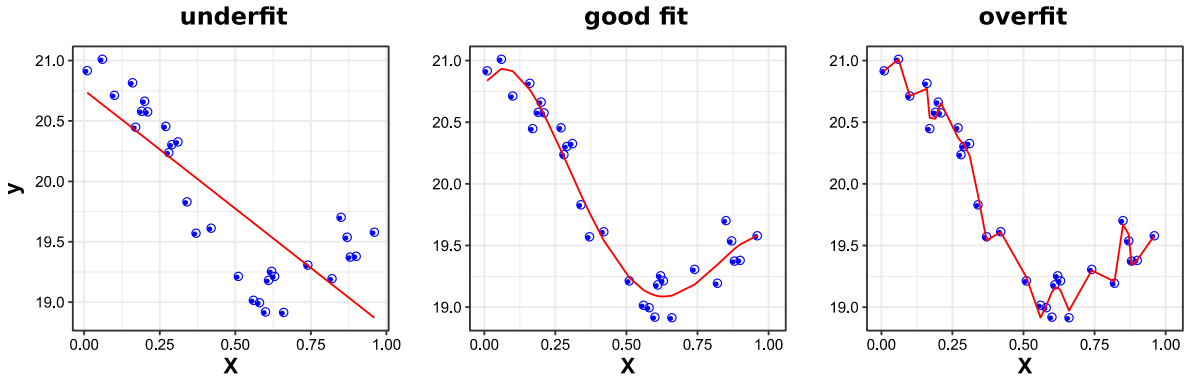

Abbildung 4: Unteranpassung *(underfit)*, gute Anpassung *(good fit)* und Überanpassung *(overfit)* beim Modelltraining einer Regression. Die blauen Punkte sind die genauen Werte des Datensatzes und die rote Linie gibt die Näherung durch das trainierte Modell an. Jeder Datenpunkt hat die beiden Eigenschaften x und y. Quelle: [14, S. 6]

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 [15]. 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 Modell gefunden werden kann. Außerdem kann das Ergebnis der Validierung als Indikator zur Bewertung der Leistungsfähigkeit eines Modells herangezogen werden. Wenn die Validierung mit einem Datensatz durchgeführt wird, welcher wie der Testdatensatz, dem Trainingsdatensatz vorenthalten wurde, spricht man von der 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 $\langle w \cdot x \rangle + b = 0$ beschrieben werden. Später kann anhand des Vorzeichens der Funktion $f(x) = \langle w \cdot x \rangle + b$ bei Einsetzung eines zu klassifizierenden Datenpunktes x, die Klasse bestimmt werden. Um die optimale Hyperebene zu finden, muss der Rand (Margin) um die Klassengrenzen möglichst breit werden *(engl. large-margin-classification)* [16] , bzw. $\frac{1}{\|w\|}$, was dem Margin entspricht, maximiert werden. Mithilfe des Sattelpunktes der Lagrange-Funktion [17] lassen sich die beiden Stützvektoren definieren, die sich bei einer Margin von ± 1 befinden und somit die Hyperebene umgeben (siehe Abbildung 5).

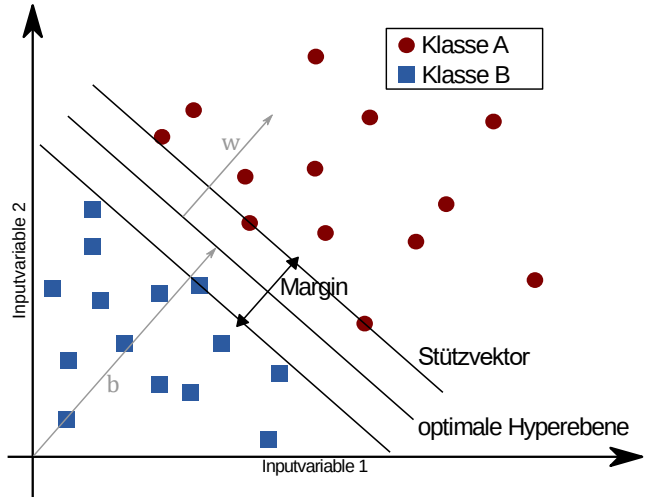

Abbildung 5: Lineare Klassifizierung eines Datensatzes mit zwei Eigenschaften pro Datenpunkt. Variablen: (Bias b und Richtungsvektor der Hyperebene w). Quelle: Eigene Darstellung

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 ξ, welche einerseits Fehleinordnungen erlaubt, diese aber andererseits auch bestraft, stellt eine Möglichkeit dar, diesen potenziellen Fehler zu vermindern. ξ 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 n-ten Dimension hat die Hyperebene genau $n - 1$ 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 [16]. 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 [18].

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

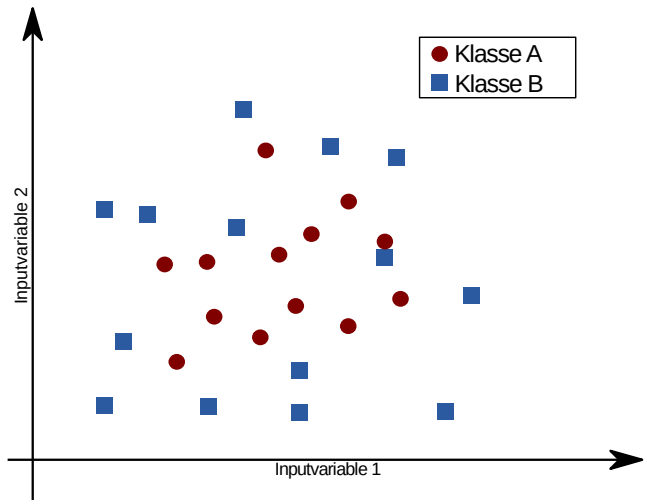

Abbildung 6: Nicht linear trennbare Daten eines Datensatzes mit zwei Eigenschaften pro Datenpunkt. Quelle: Eigene Darstellung

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 [19]. 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 Neuronen 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 [20]. 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 [21]. 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.

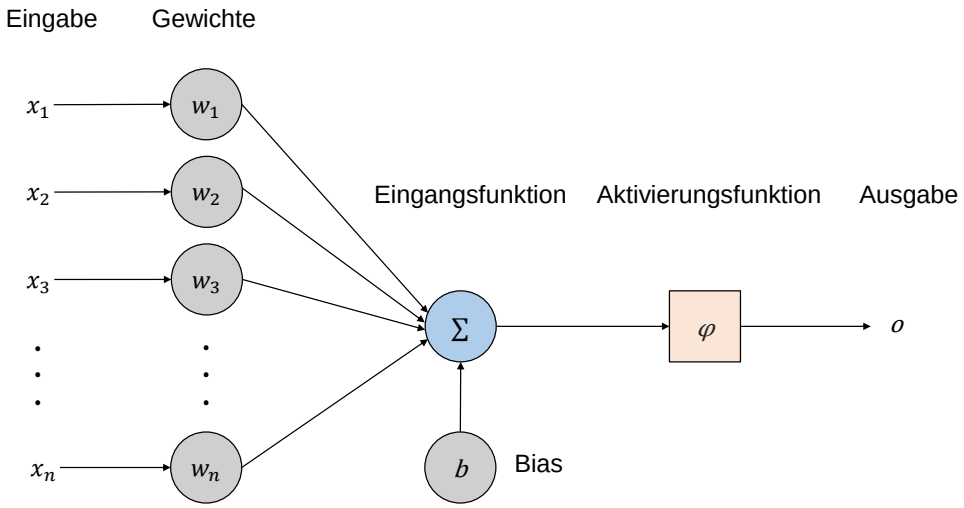

Abbildung 7: Einlagiges Perzeptron. Quelle: Eigene Darstellung

Jedem Eingangssignal $x_i \in \mathbb{R}$ des Perzeptrons wird ein Gewicht $w_i \in \mathbb{R}$ mit $i \in \{1, \dots, n\}$ zugeordnet. In der Eingangsfunktion σ wird die gewichtete Summe über die gesamte Eingabe gebildet. Zusätzlich wird auf diese Summe der Bias b addiert. Der Bias gibt den Grenzwert an, ab dem das Perzeptron feuern soll.

$$\sigma = \sum_{i=1}^{n} x_i \cdot w_i + b \tag{1.1}$$

Mithilfe einer Aktivierungsfunktion φ wird die Ausgabe $o \in \mathbb{R}$ des Perzeptrons bestimmt:

$$o = \varphi(\sigma) \tag{1.2}$$

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

$$f_\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.3}$$

Bei der Sigmoidfunktion handelt es sich um eine differenzierbare, nicht lineare Funktion. Sie führt, verglichen mit einer einfachen Schwellenwertfunktion, zu einer weicheren Aktivierung des Neurons [22]. 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.

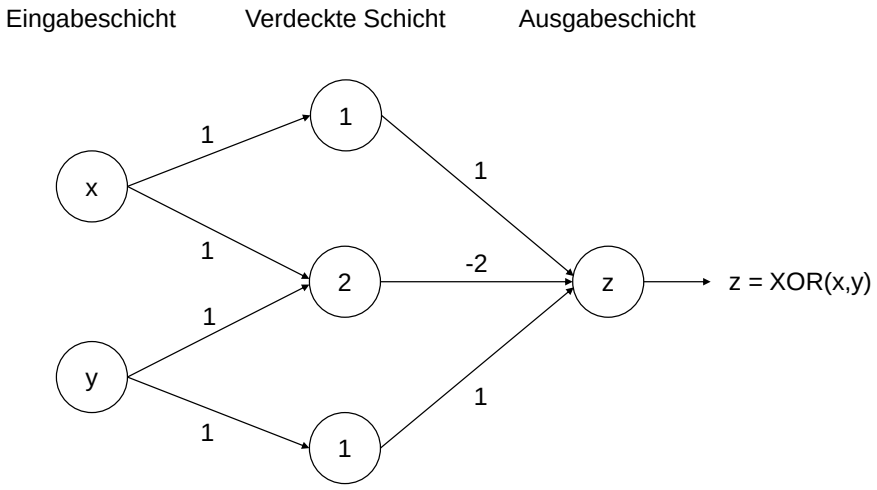

Abbildung 8: Mehrlagiges XOR-Netz. Achtung, hier stellt jeder Kreis ein komplettes Perzeptron dar (wichtiger Unterschied zu Abbildung 7). x und y stehen für die Eingabeoperanden 1 (wahr) und 0 (falsch). Auf den Pfeilen ist je das Gewicht angegeben. Die Neuronen der verdeckten Schicht besitzen zudem einen 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. Z ist die Variable für die gewichtete Summe der Ausgabewerte des Netzes. Quelle: Eigene Darstellung nach [23, S. 4]

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

1) $x = y = 0 \rightarrow z = 1 \cdot 0 - 2 \cdot 0 + 1 \cdot 0 = \mathbf{0}$: Kein Perzeptron feuert.
2) $x = y = 1 \rightarrow z = 1 \cdot 1 - 2 \cdot 1 + 1 \cdot 1 = \mathbf{0}$: Alle Perzeptren feuern.

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

1) $x = 1;\ y = 0 \rightarrow z = 1 \cdot 1 - 2 \cdot 0 + 1 \cdot 0 = \mathbf{1}$: Nur das obere Perzeptron feuert.
2) $x = 0;\ y = 1 \rightarrow z = 1 \cdot 0 - 2 \cdot 0 + 1 \cdot 1 = \mathbf{1}$: 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 anderen 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 Neuronen der i-ten verdeckten Schicht die Ausgabe der Neuronen der $i-1$-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 Neuronen 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. 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 [24]. Das Training kann auch als Optimierungsproblem aufgefasst werden, bei dem die Gewichte der einzelnen Neuronen 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 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.

1.2 Pharmazeutische Datenflut im Zeitalter von Big Data

Die Wirkung von vieler Medikamente beruht heutzutage auf der Interaktion eines Wirkstoffes mit einem Zielmolekül. In den aller meisten Fällen handelt es sich bei dem Wirkstoff um eine niedermolekulare Verbindung *(engl. small molecule)*, d.h. eine chemische Verbindung mit einem geringen Molekulargewicht und bei den Zielmolekülen um Proteine wie z.B. Enzyme, Rezeptoren oder Ionenkanäle. Es besteht ein pharmazeutisches Interesse zu wissen, zwischen welchen niedermolekularen Verbindungen und welchen Proteinen eine Interaktion besteht. Anhand dieser Daten lassen sich Aussagen zu der Wirksamkeit und zu möglichen Nebenwirkungen von Wirkstoffkandidaten treffen.

Es existieren bereits diverse Moleküldatenbanken, wie ChEMBL, DrugBank oder PDBbind [1] [2] [3], in denen Informationen zu der Interaktion von kleinen chemischen Verbindungen mit Proteinen *(engl. compound-protein interactions; CPI)* gesammelt und dargestellt werden. 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 großen Datenflut ist es erschwert, alle neuen Erkenntnisse zur Wirkung von Arzneistoffen aus aktuellen wissenschaftlichen Arbeiten in die Datenbanken zu integrieren. Infolgedessen werden Informationen übersehen und sind dadurch für die Datenbanken verloren. Automatisierte und computergestützte Techniken bieten einen Ansatzpunkt, um der Datenflut gerecht zu werden.

Der Begriff *Text Mining* beschreibt die Menge an Techniken zum automatischen Extrahieren von neuen und zuvor unbekannten Informationen aus unstrukturierten Texten [26, S. 42]. Diese Techniken verwenden häufig Algorithmen, die auf maschinellem Lernen basieren. 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, 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.

1.3 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).

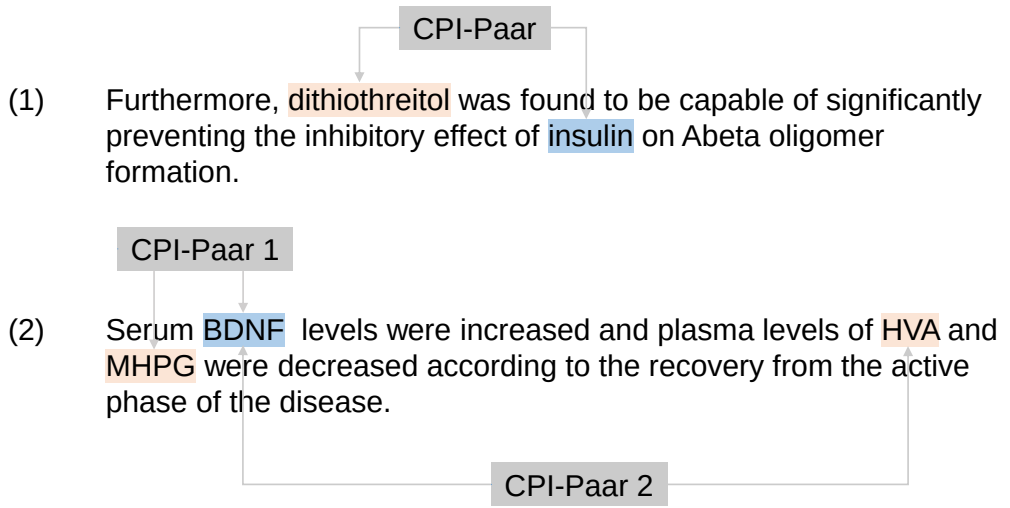

Abbildung 9: Beispielsätze mit CPI-Paaren aus wissenschaftlichen Arbeiten. Die vorhandenen Proteine sind mit blau und die kleinen chemischen Verbindungen mit rot hinterlegt. **(1)** Beispielsatz: Hier besteht eine funktionale Beziehung (d.h. Interaktion) zwischen dem Protein „insulin“ und der Verbindung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

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.

1.3.1 Historie von Beziehungs-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. et al.* zur Extraktion von CPI-Paaren aus der Literatur verwendet [36]. 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].

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. et al.* aus dem Jahr 2020 [36] wurden die beiden Kern-Methoden *all-paths graph (APG)* und *shallow linguistic (SL)* 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 APG Kern-Methode wird ein Abhängigkeits-Graph der Satzstruktur verwendet, um die Ähnlichkeiten mit Hilfe von gewichteten Pfaden zu bestimmen. Die SL 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 (c,n und w) angegeben. Für eine Beschreibung der einzelnen Werte siehe 3.1.4. Quelle: Eigene Darstellung nach [36]

Kern-Methode	Sensitivität	Spezifität	Genauigkeit	F-Maß	AUC
APG	0,817	0,718	0,766	0,790	0,846
SL	0,795	0,702	0,750	0,772	0,825
kombiniert	0,685	0,816	0,805	0,740	-

Der verwendete Datensatz wurde aus den Abstracts der 40.000 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 Interaktion 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.

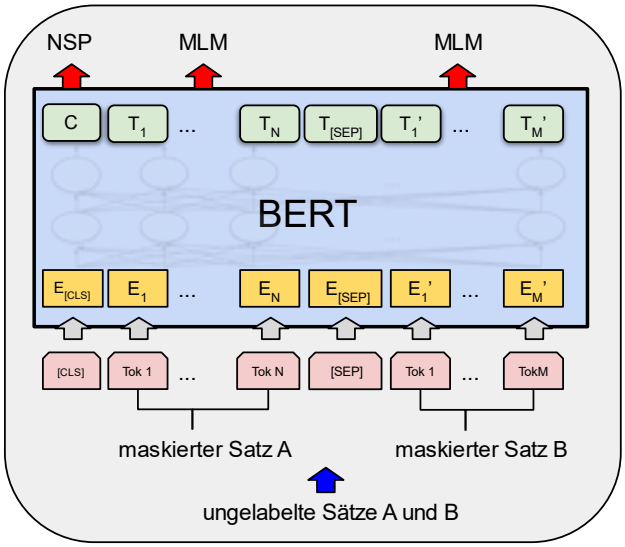

Abbildung 10: Schemata zum Vortraining von BERT. Die beiden Sätze A und B werden in die Token Tok aufgeteilt und manche Token maskiert. Die Token werden anschließend mittels Worteinbettung in eine Vektorform E überführt und diese dem Kodierer für NSP und MLM zugeführt. Als Output gibt das neuronale Netz die binäre Variable C als Ergebnis des NSP und T als Vorhersage des maskierten Tokens an. Quelle: Eigene Darstellung nach [38, S. 3]

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 Abstracts (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 Methoden verwendet, um das BioBERT-Modell auf das CPI-Text Mining vorzubereiten.

2 Zielsetzung der Arbeit

Diese Bachelorarbeit soll, als Vergleich von Kern-Methoden mit einem auf tiefes Lernen basierenden Ansatz zum Klassifizieren von CPI-Paaren 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 Klassifizierung.

3 Methoden und Datensätze

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

3.1 Grundlagen

3.1.1 Systemeigenschaften

Betriebssystem:	Ubuntu 18.04.5 LTS (Bionic Beaver)[1]
Prozessor:	Intel® Core™ i5-6600K CPU @ 4,20GHz
Grafikkarte:	NVIDIA® GeForce® GTX 1060 6GB
Programmiersprache:	Python 3.6.9

3.1.2 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]. 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 letzten Endes zum Einsatz kam, ist auf GitHub in einem neuen Repository verfügbar.[3]

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].

---

[1] Zuerst wurde in einer VirtualBox unter Windows 10 begonnen, dann mittels einer neuen SSD ein Dual-Boot-System mit Ubuntu eingerichtet.

[2] https://github.com/dmis-lab/biobert

[3] https://github.com/mmmddd98/ba_git

[4] https://github.com/dmis-lab/biobert-pytorch

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 in dieser Arbeit verwendeten Versionen, können der Datei `requirements.txt` entnommen 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
```

3.1.3 Verwendeter Datensatz

Der dieser Arbeit zu Grunde liegende Datensatz stammt aus der Publikation von *Döring K. et al.* [36]. Die Struktur der XML-Datei lässt sich Abbildung 11 entnehmen.

```
</document>
<document id="DS.d38" origId="18198997">
    <sentence id="DS.d38.s0" origId="18198997-256" text="Butyrate enemas upregulate Muc genes
     expression but decrease adherent mucus thickness in mice colon.">
        <entity id="DS.d38.s0.e0" origId="264" charOffset="0-8" type="compound" text="Butyrate"/>
        <entity id="DS.d38.s0.e1" origId="P0A9M0" charOffset="27-30" type="protein" text="Muc"/>
        <pair e1="DS.d38.s0.e0" e2="DS.d38.s0.e1" id="DS.d38.s0.i0" interaction="True"/>
    </sentence>
    <sentence id="DS.d38.s1" origId="18198997-257" text="We demonstrated that butyrate stimulated
     the gene expression of both secreted (Muc2) and membrane-linked (Muc1, Muc3, Muc4) mucins.">
        <entity id="DS.d38.s1.e0" origId="264" charOffset="21-29" type="compound" text="butyrate"/>
        <entity id="DS.d38.s1.e1" origId="Q62635" charOffset="79-83" type="protein" text="Muc2"/>
        <entity id="DS.d38.s1.e2" origId="Q02496" charOffset="106-110" type="protein" text="Muc1"/>
        <pair e1="DS.d38.s1.e0" e2="DS.d38.s1.e1" id="DS.d38.s1.i0" interaction="True"/>
        <pair e1="DS.d38.s1.e0" e2="DS.d38.s1.e2" id="DS.d38.s1.i1" interaction="True"/>
    </sentence>
```

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 chemische 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“). Quelle: Eigene Darstellung

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

Tabelle 2: Inhalt des CPI-DS Quelle: Eigene Darstellung nach [36, S.6]

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

3.1.4 Parameter zur Bewertung von Modellen

Wenn ein maschinelles Lern 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 Klassifizierungsaufgabe mit den Klassen *Positiv* und *Negativ*, vier mögliche Ausgänge der Vorhersage der Klasse:

1) Richtig positiv TP *(engl. true positiv)*: Richtige Vorhersage der Klasse *Positiv*
2) Falsch positiv FP *(engl. false positiv)*: Falsche Vorhersage der Klasse *Positiv*
3) Richtig negativ TN *(engl. true negativ)*: Richtige Vorhersage der Klasse *Negativ*
4) Falsch negativ FN *(engl. false negativ)*: Falsche Vorhersage der Klasse *Negativ*

Daraus lassen sich die folgenden Parameter ableiten:

Sensitivität

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

$$R = \frac{TP}{TP + FN} \tag{3.1}$$

Spezifität

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

$$S = \frac{TN}{TN + FP} \tag{3.2}$$

Genauigkeit

Die Genauigkeit P *(engl. precision)* gibt die Wahrscheinlichkeit an, dass ein als *Positiv* klassifiziertes Objekt auch wirklich *Positiv* (TP) 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 (FN). Durch eine Verbesserung der Genauigkeit wird die Anzahl an FP verringert. Das macht sie für den Vergleich von CPI Klassifizierungsmodellen von Bedeutung.

$$P = \frac{TP}{TP + FP} \tag{3.3}$$

F-Maß

Das F-Maß, ein kombiniertes Maß, ist das harmonische Mittel von Genauigkeit P und Sensitivität S. Mit einem Wert zwischen 0 und 1 wird beschreiben, wie gut die Leistung des Modells allgemein ist. Daher wird das F-Maß oft herangezogen, wenn es darum geht, mehrere Modelle miteinander zu vergleichen. Ein guter Wert für das F-Maß liegt nahe bei 1 und bedeutet, dass die Zahl an falschen Klassifizierungen (FP und FN) relativ gering ist.

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.4}$$

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 R und dem Komplement der Spezifität $1 - S$. 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.

3.2 Datensatzpräparation

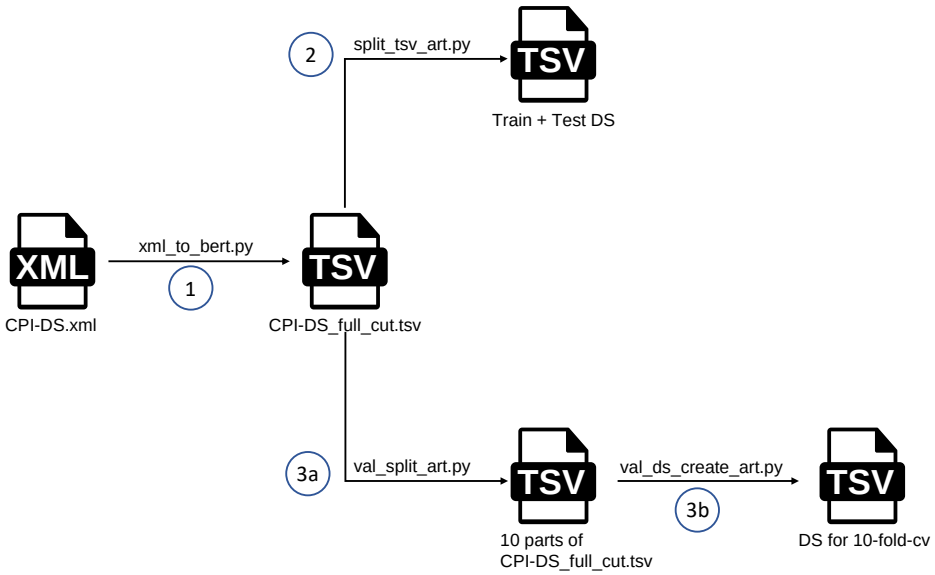

Abbildung 12: Übersicht über die gesamte Arbeit mit dem Datensatz und den dafür verwendeten Python-Skripten. **Schritt 1** entspricht der Datensatzpräparation. In **Schritt 2** wird der Datensatz zu einem Trainings- und Testdatensatz mit beliebiger Größe gesplittet (siehe 3.4.1). Mit **Schritt 3a und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

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 ist in den Ergebnissen unter Abbildung 16 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 gekürzte Version des Datensatzes erzeugt werden soll. Dies ist notwendig, um Probleme bei dem Ersetzen der Entitäten mit ihren Platzhaltern zu verhindern. Beim Kürzen werden 13 Sätze mit CPI-Paaren aussortiert, bei denen sich das Protein mit der chemischen Verbindung überschneidet. Die aussortierten Sätze können durch Bestätigen der Frage "aussortierte Sätze printen?" im Terminal ausgegeben werden.

Das Splitten dieses konvertierten Datensatzes für das Trainieren von BioBERT ist in 3.3.2 und 3.4.1 beschrieben.

3.3 10-fache Kreuzvalidierung

3.3.1 *k*-fache Kreuzvalidierung

Bei einer k-fachen Kreuzvalidierung wird der Datensatz zufällig in k Teilmengen der gleichen Größe gesplittet. Ein Modell wird jeweils mit $k - 1$ Teilmengen trainiert (Trainingsdatensatz) und anschließend an der übrigen Teilmenge als Testdatensatz geprüft. Die erhaltenen Bewertungen jedes der k Durchgänge werden anschließend gemittelt. Dadurch wird erreicht, dass der Einfluss von einfacheren und komplexeren 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 10-fache Kreuzvalidierungen durchgeführt.

3.3.2 Splitten des Datensatzes

Anhand von Abbildung 13 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.

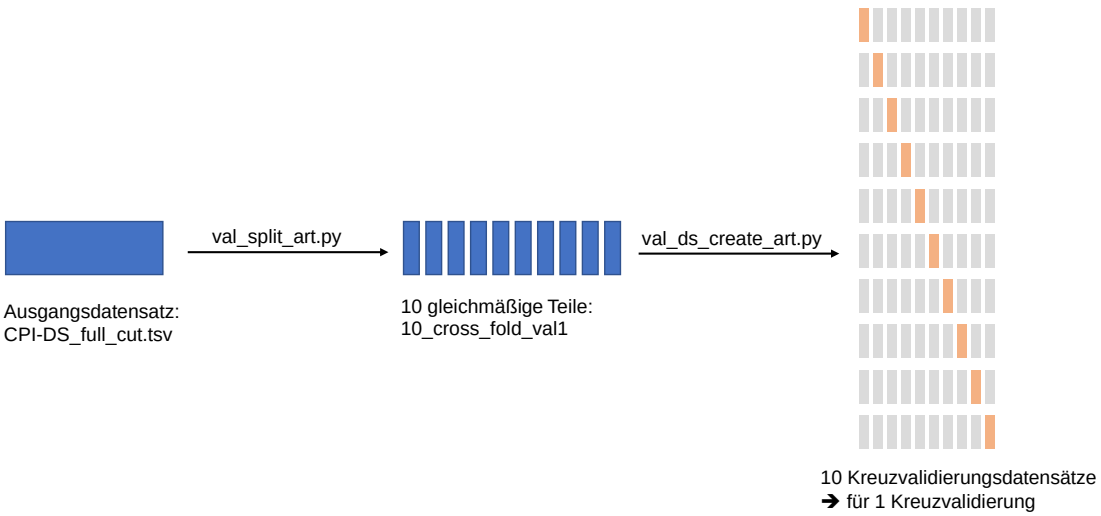

Abbildung 13: Erzeugen der Kreuzvalidierungsdatensätze. Diese Schritte entsprechen 3a und 3b in Abbildung 12. In grau sind die Teile des Ausgangsdatensatzes dargestellt, welche jeweils zum Trainingsdatensatz zusammengefasst werden. Die orangenen Abschnitte stehen für den Testdatensatz. Für die zweite durchgeführte Kreuzvalidierung werden zehn neue Teile des Ausgangsdatensatzes erzeugt. Quelle: Eigene Darstellung

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 den Benutzer, wie viele 10-fache Kreuzvalidierungsdatensätze benötigt werden und erstellt diese.

In Abbildung 14 ist die `main()` Funktion des Skriptes dargestellt. Es lässt sich in drei funktionale Teile unterteilen:

1) **Einlesen des Datensatzes:** Es wird 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 Split 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 die Standardabweichung. Es wurde für die zehn erzeugten Teil-Datensätze jeweils die Anzahl der enthaltenen Interaktionen und Sätze bestimmt. Davon wird die SD berechnet und mit dem Grenzwert verglichen. Um das Verhältnis der Anzahl von funktionalen zu nicht funktionalen Interaktionen eines Teildatensatzes zu bestimmen, werden die beiden Anzahlen für alle Teil-Datensätze bestimmt und der Quotient $\frac{Anzahl\ nicht\ funktionaler\ Interaktionen}{Anzahl\ funktionaler\ Interaktionen}$ 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

3) **Speichern der Teil-Datensätze:** Die zehn Teil-Datensätze werden gespeichert. Zusätzlich wird für jeden Teil eine Datei `results.tsv` erstellt, in welcher statistische Daten zu dem Teil-Datensatz angegeben sind.

```
def main(file,ip_index_spalte,run):
        parts_number = 10
        os.mkdir(str(parts_number) + "_cross_fold_val" + str(run))
        file_lists = read_file(file)
        sentence_list_full = file_lists[0]
        label_list_full = file_lists[1]
        sentence_id_list_full = file_lists[2]
        sentence_id_list_full_sorted = collections.OrderedDict.fromkeys(sentence_id_list_full).keys()
        anzahl_satz_ges = len(sentence_id_list_full_sorted)
        print("anzahl der sätze:",anzahl_satz_ges)
        art_id_list_full = convert_sen_to_art(sentence_id_list_full)
        art_id_list_full_sorted = collections.OrderedDict.fromkeys(art_id_list_full).keys()
        anzahl_art_ges = len(art_id_list_full_sorted)
        print("anzahl der art:",anzahl_art_ges)
        file_tupel_full = lists_to_tuple_list(file_lists)
        anzahl_int_ges = len(file_tupel_full)
        print("anzahl der interactions:",anzahl_int_ges)
        print("1.Versuch")

        random_zahlen_part = get_random_part_neu_neu(art_id_list_full_sorted, parts_number)
        list_of_art_lists = create_split_id_lists(random_zahlen_part,art_id_list_full_sorted)
        tuple_split = split_tuples(file_tupel_full,list_of_art_lists)
        c = 2
        while auswerten(tuple_split,22,0.12,10) == False:
                print(str(c) + ".Versuch")
                random_zahlen_part = get_random_part_neu_neu(art_id_list_full_sorted, parts_number)
                list_of_art_lists = create_split_id_lists(random_zahlen_part,art_id_list_full_sorted)
                tuple_split = split_tuples(file_tupel_full,list_of_art_lists)
                c = c + 1

        save_all(tuple_split,file,parts_number,ip_index_spalte,list_of_art_lists,anzahl_int_ges,…,…)
        print(">>> Splitten in die " + str(parts_number) + " Parts erfolgreich")
```

Abbildung 14: `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. Quelle: Eigene Darstellung

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 Split als Kommandozeilenargument angegeben. 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.

3.3.3 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 (*learning_rate* in BioBERT) 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 angegeben:

```
$ cd Biobert/tensorflow/results/cross_val
$ python3 val_create_results.py 10_cross_fold_val1
```

3.3.4 Reproduzieren der Kreuzvalidierung 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. et al.* zu erreichen, werden die 10-fachen-Kreuzvalidierungen der Kern-Methoden wiederholt. Die genaue Implementierung ist in einem GitHub Repository veröffentlicht.[5] Nach dem Installieren der notwendigen Pakete, werden die beiden Kreuzvalidierungen, wie in Punkt „How to run the Kernel Pipelines“[6] beschrieben, durchgeführt. Eine wichtige Änderung zu der originalen Anleitung ist, dass die Skripte `Run_APG_Kernel.py` und `Run_SL_Kernel.py` so modifiziert wurden, dass nur die Kreuzvalidierungen mit den Parametern der Kern-Methode wiederholt werden, welche in der Publikation die besten Ergebnisse gezeigt haben. Dafür werden in `Run_APG_Kernel.py` Zeile 405 mit „Training(-2,-1)“ und Zeile 234 in `Run_SL_Kernel.py` mit „Training(1,2, 3,4)“ überschrieben. Der verwendete Datensatz ist `CPI-DS.xml` mit der Besonderheit, dass dieselben 13 Sätze, wie in 3.2 beschrieben von Hand aussortiert wurden.

3.4 Einfluss der Trainingsdatensatzgröße

Um zu untersuchen, wie sich die Leistung des Modells bei einem größeren, 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, bestehend aus je einem Trainingsdatensatz und einem 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.

[5] https://github.com/KerstenDoering/CPI-Pipeline

[6] https://github.com/KerstenDoering/CPI-Pipeline#how-to-run-the-kernel-pipelines

3.4.1 Splitten des Datensatzes

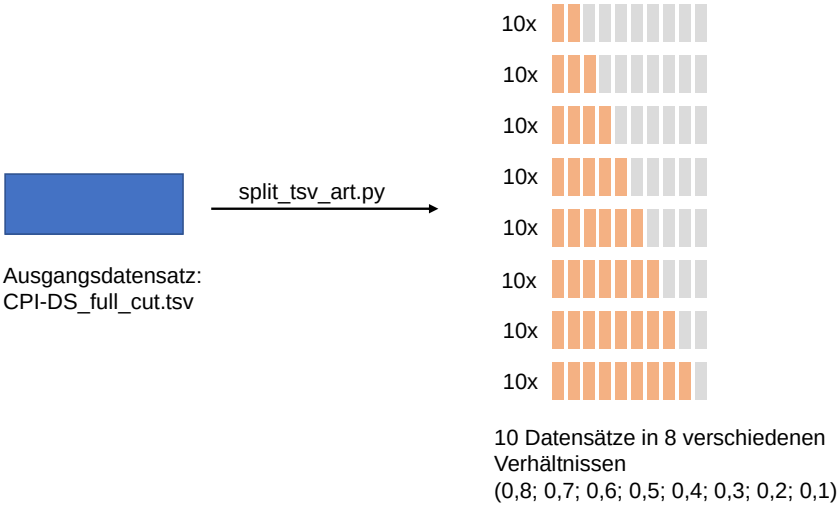

Abbildung 15: Erzeugen der Datensätze für das Untersuchen des Einflusses der Trainingsdatensatzgröße. Die Teile des Trainingsdatensatzes sind in grau und die des Testdatensatzes in orange dargestellt. Quelle: Eigene Darstellung

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 Indexspalten benötigt werden und kann selbst, durch Eingabe in das Terminal, den benötigten Anteil des Testdatensatzes am Ausgangsdatensatz, sowie die gesamte Anzahl mit diesem Split erzeugter 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 zum Beispiel sein, dass alle Artikel mit vielen beschriebenen Interaktionen im Testdatensatz landen und dadurch das Verhältnis der 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 neue zufällige Splits so lange erzeugt, bis bestimmte Grenzwerte eingehalten werden (siehe Tabelle 4). Die Ergebnisse dieser Bewertung werden für alle erfolgreichen Splits 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 $q = \frac{Anzahl\ nicht\ funktionaler\ Interaktionen}{Anzahl\ funktionaler\ Interaktionen}$ für den Test- und Trainingsdatensatz gebildet. Es wird die relative Abweichung dieses Quotienten $\frac{q_{test}}{q_{training}}$ zu dem des Ausgangsdatensatzes berechnet. Ein Split wird gespeichert, wenn für alle drei Merkmale die maximale relative Abweichung unterschritten wird.

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

3.4.2 Training des Modells und Vorhersagen

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 des 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 gleichen der für die Validierung verwendeten Funktion. 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 arithmetische 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 anzugeben.

4 Ergebnisse und Diskussion

In diesem Kapitel werden die Ergebnisse präsentiert und diskutiert. Alle Rohdaten und Ergebnisse sind in dem Ordner `Data/results` in dem GitHub Repository digital zu finden.

4.1 Konvertierter Ausgangsdatensatz

Der grundlegende Schritt ist die Konvertierung des ursprünglichen Datensatzes vom XML-Format zu dem in Abbildung 16 gezeigten im TSV-Format. Der konvertierte Datensatz wird für die Kreuzvalidierung und das Ermitteln des Einflusses der Trainingsdatensatzgröße auf die Klassifizierungsleistung verwendet.

sentence	label	sentence-id
@PROTEIN$ enables Ca2+-activated @COMPOUND$-conductance in epithelia.	1	17003041-302
Calcium-dependent Cl (-) currents were activated by @COMPOUND$ in HEK293 cells expressing	0	17003041-304
Calcium-dependent @COMPOUND$ (-) currents were activated by ATP in HEK293 cells expressi	0	17003041-304
The absorption spectrum of the @PROTEIN$ enzyme showed an absorption peak at 425nm indi	1	17321121-703
When Vitreoscilla were grown in medium containing 60mM @COMPOUND$ under both normal	1	17403602-106
@COMPOUND$) suppresses beta-amyloid-induced neurotoxicity through inhibiting c-Abl/FE65	1	17590240-123
@COMPOUND$) suppresses beta-amyloid-induced neurotoxicity through inhibiting c-@PROTEI	1	17590240-123
@COMPOUND$) suppresses beta-amyloid-induced neurotoxicity through inhibiting c-Abl/@PR	1	17590240-123
Here, we used a human neuronal cell line MC65 conditional expression of an amyloid precursor	0	17590240-123
Here, we used a human neuronal cell line MC65 conditional expression of an amyloid precursor	0	17590240-123
The @PROTEIN$ was purified to homogeneity using ammonium @COMPOUND$ precipitation, a	0	17616381-938
The @PROTEIN$ was purified to homogeneity using @COMPOUND$ sulfate precipitation, and i	0	17616381-938
In addition, we demonstrated that Hsp20, @PROTEIN$ and HspB8 induced interleukin-6 produc	0	17629591-228
In addition, we demonstrated that Hsp20, HspB2 and HspB8 induced @PROTEIN$ production in	1	17629591-228
In addition, we demonstrated that Hsp20, HspB2 and @PROTEIN$ induced interleukin-6 produc	0	17629591-228
Furthermore, @COMPOUND$ was found to be capable of significantly preventing the inhibitory	1	17692997-193
The scavenger receptor, class B, type I (@PROTEIN$) is critical in maintaining the homeostasis o	1	17719144-117
The scavenger receptor, class B, type I (@PROTEIN$) is critical in maintaining the homeostasis o	1	17719144-117
SR-BI binds @PROTEIN$ (HDL) and mediates the selective transfer of cholesteryl esters and @C	0	17719144-117
@PROTEIN$ binds high-density lipoproteins (HDL) and mediates the selective transfer of chole	1	17719144-117
Thus, @PROTEIN$ influences neural and cognitive processes, a finding that highlights the contr	0	17719144-117
Thus, @PROTEIN$ influences neural and cognitive processes, a finding that highlights the contr	0	17719144-117
@COMPOUND$ is believed to confer its diabetogenic effect by inhibiting pancreatic @PROTEIN	1	17768029-984
@COMPOUND$ supplementation prevents the exercise-induced reduction of serum paraoxona	1	17882129-602
Although @PROTEIN$4 has been repeatedly associated with altered sphingomyelin and @COM	1	17888544-779
@PROTEIN$ modified the association between cholesterol and cognitive decline, and the asso	1	17888546-781
@PROTEIN$ modified the association between @COMPOUND$ and cognitive decline, and the a	1	17888546-781
Here we show that during growth in media containing @COMPOUND$ and in complex medium	0	17890844-955
The mood stabilizers @COMPOUND$ and valproate selectively activate the promoter IV of @PF	1	17925795-103
The mood stabilizers lithium and @COMPOUND$ selectively activate the promoter IV of @PRO	1	17925795-103

Abbildung 16: 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 die identische ID. Quelle: Eigene Darstellung

4.2 Kreuzvalidierung mit Modellvergleich

In 4.2 werden erst die Ergebnisse der durchgeführten Kreuzvalidierungen präsentiert, dann für sich und anschließend vergleichend diskutiert.

4.2.1 Kreuzvalidierungen BioBERT

Für die Ermittlung der Klassifizierungsleistung von BioBERT auf den `CPI-DS_full_cut.tsv` Datensatz wurden zwei 10-fache-Kreuzvalidierungen durchgeführt. In Tabelle 5 und Tabelle 6 sind die Ergebnisse der zehn einzelnen Teil-Datensätze ausführlich dargestellt.

Tabelle 5: Ergebnisse der ersten Kreuzvalidierung

Teil-Datensatz	**Sensitivität**	**Spezifität**	**Genauigkeit**	**F-Maß**	**Laufzeit** [min]
Ds-1	0,908	0,724	0,789	0,844	73
Ds-2	0,903	0,779	0,823	0,861	68
Ds-3	0,874	0,815	0,829	0,851	68
Ds-4	0,945	0,759	0,832	0,885	67
Ds-5	0,883	0,836	0,827	0,854	67
Ds-6	0,821	0,778	0,824	0,823	68
Ds-7	0,842	0,776	0,823	0,832	68
Ds-8	0,849	0,705	0,772	0,809	68
Ds-9	0,873	0,859	0,860	0,866	79
Ds-10	0,842	0,787	0,813	0,827	78
Mittelwert	**0,874**	**0,782**	**0,819**	**0,845**	**70**
Standardabweichung	0,037	0,047	0,024	0,023	4

Tabelle 6: Ergebnisse der zweiten Kreuzvalidierung

Teil-Datensatz	**Sensitivität**	**Spezifität**	**Genauigkeit**	**F-Maß**	**Laufzeit** [min]
Ds-1	0,885	0,816	0,825	0,854	73
Ds-2	0,909	0,770	0,799	0,851	70
Ds-3	0,861	0,775	0,845	0,853	70
Ds-4	0,907	0,869	0,902	0,904	69
Ds-5	0,830	0,835	0,842	0,836	68
Ds-6	0,830	0,795	0,810	0,820	68
Ds-7	0,880	0,788	0,823	0,850	68
Ds-8	0,837	0,888	0,894	0,864	68
Ds-9	0,832	0,815	0,849	0,841	69
Ds-10	0,905	0,738	0,763	0,828	68
Mittelwert	**0,868**	**0,809**	**0,835**	**0,850**	**69**
Standardabweichung	0,034	0,046	0,042	0,023	2

Die aussagekräftigen Werte aus den Tabellen sind jeweils ausschließlich die Mittelwerte der beiden Kreuzvalidierungen. Alle Ergebnisse der Teil-Datensätze unterliegen Schwankungen, welche damit zu erklären sind, dass sich der Trainingsdatensatz immer aus unterschiedlichen Teil-Datensätzen zusammensetzt (siehe 4.2.2). Als F-Maß wurde ein Wert von 0,845 bzw. 0,850 bestimmt. Insgesamt decken sich die anderen ermittelten Parameter zur Bewertung der Leistungsfähigkeit des Modells, wie z.B. die Genauigkeit, im Vergleich der beiden Kreuzvalidierungen ebenfalls gut. Abgesehen von dem unterschiedlichen zufälligen Splitten des Datensatzes in die Teil-Datensätze, ist das zufällige Setzen der Gewichte vor dem Training der Modelle ein wichtiger Punkt, um generelle Abweichungen der Parameter zwischen den Kreuzvalidierungen zu erklären. Dadurch erfahren die Gewichte der Modelle im Verlauf des Trainings unterschiedliche Anpassungen, sodass sich die fertigen Modelle in ihren Gewichtungen der einzelnen Perzeptren unterscheiden. Das resultiert in den leicht unterschiedlichen Vorhersagen beim Testen (siehe Tabelle 5 und Tabelle 6).

Die gesamte Laufzeit der ersten Kreuzvalidierung beträgt 11 h 48 min und die der zweiten 11 h 31 min. Für den späteren Vergleich der Modelle hat nur die komplette Zeit einer Kreuzvalidierung Aussagekraft. Nahezu die komplette Laufzeit entfällt auf das Training des Modells und nur wenige Minuten auf das Klassifizieren der CPI-Paare des Testdatensatzes. Für das rechenlastige Training wird die CPU erheblich beansprucht. So beeinflussen im Umkehrschluss auch andere Prozesse, die während des Trainings ebenfalls die CPU beanspruchen, die Laufzeit. Die Kreuzvalidierungen wurden jeweils am Abend gestartet und über Nacht laufengelassen. Bei der ersten Kreuzvalidierung können so die deutlich erhöhten Laufzeiten der Teildatensätze Ds-1, Ds-9 und Ds-10 auf eine anderweitige CPU-Auslastung zurückzuführen sein. Es ist es sehr wahrscheinlich, dass der prozentuale Unterschied der Laufzeiten beider Kreuzvalidierungen (2,5 %), unter den CPU-Auslastungs-bedingten Schwankungen liegt.

4.2.2 Homogenität der verwendeten Teil-Datensätze

Der Ausgangsdatensatz stammt direkt aus Publikationen des Jahres 2009. Das bedeutet, dass die in diesem enthaltenen CPI-Paare, die Komplexität und das Verhältnis von funktionalen zu nicht funktionalen Interaktionen aus der Literatur bestens abbilden. Beim Splitten des Datensatzes in die für die Kreuzvalidierung benötigten Teil-Datensätze wurde versucht diese Merkmale in alle Teildatensätze zu übertragen. Um das zu erreichen, wurde beim Splitten, die in Tabelle 3 angegebenen Grenzwerte, eingehalten. Die Idee ist, dass jeder Teil-Datensatz den kompletten Datensatz repräsentieren kann. Dadurch spiegeln die Ergebnisse der Kreuzvalidierung besser die Leistung des Modells, in Hinblick auf die zukünftige Anwendung in der Literatur, wider.

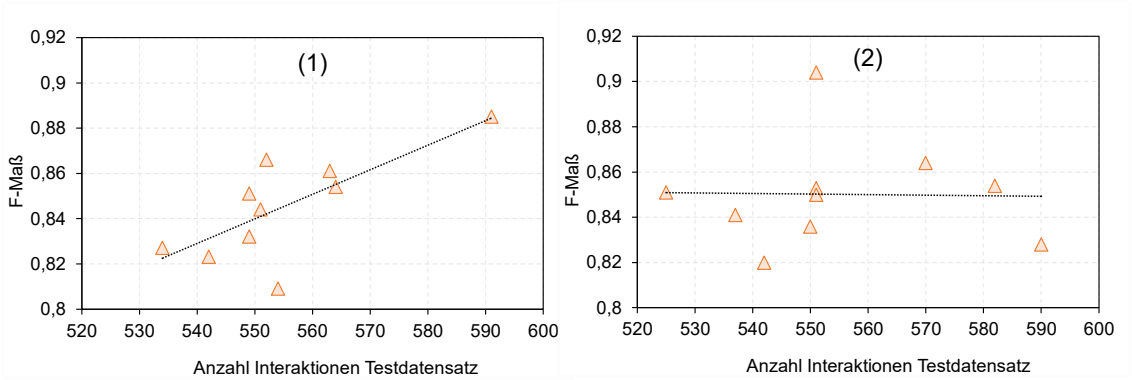

Abbildung 17: Zusammenhang von F-Maß und Anzahl der Interaktionen der zehn Teil-Datensätze von Kreuzvalidierung 1(siehe **(1)**) und Kreuzvalidierung 2 (siehe **(2)**). Quelle: Eigene Darstellung

In Abbildung 17 lässt sich für die Teil-Datensätze der zweiten Kreuzvalidierung kein Zusammenhang der Leistung (hier auf das F-Maß reduziert) und der Anzahl der Interaktionen erkennen. Das bedeutet, dass bei der zweiten Kreuzvalidierung die Teil-Datensätze gut den Ausgangsdatensatz abbilden. Bei der ersten Kreuzvalidierung ist ein leichter Trend zu erkennen (quadrierter Pearson-Korrelationskoeffizient von 0,540). Daher werden für den Modellvergleich in 4.2.4 ausschließlich die Mittelwerte über die Teil-Datensätze der beiden Kreuzvalidierungen von BioBERT herangezogen (siehe Tabelle 5 und Tabelle 6). Im Mittelwert eliminieren sich die Schwankungen durch die Inhomogenität der Teildatensätze gegenseitig.

4.2.3 Reproduzierte Kern-Methoden Ergebnisse

Die Resultate der 10-fachen-Kreuzvalidierungen der beiden Kern-Methoden können Tabelle 7 entnommen werden.

Tabelle 7: Ergebnisse 10-fache-Kreuzvalidierung der Kern-Methoden mit dem gekürzten Datensatz

Kern-Methode	Sensitivität	Spezifität	Genauigkeit	F-Maß	AUC
APG	0,827	0,705	0,761	0,792	0,847
SL	0,803	0,700	0,754	0,777	0,820

Wichtig ist, dass diese nicht mit dem ursprünglichen XML-Datensatz, wie in der Publikation beschrieben [36], sondern dem gekürzten Datensatz erzielt wurden. Da beim Kürzen nur 13 beschriebene Sätze entfernt wurden, sind bei den Ergebnissen, verglichen mit der Publikation (siehe Tabelle 1) keine Auswirkungen zu erkennen. Ebenfalls wird eine leicht höhere Leistung bei der APG Kern-Methode festgestellt. Die ermittelte Genauigkeit liegt mit 0,761 knapp unter dem in der Publikation bestimmten Wert. Das bestimmte F-Maß von 0,792 übersteigt das F-Maß der Publikation minimal. Die Laufzeiten für jeweils eine Kreuzvalidierung mit den beiden Kern-Methoden betragen 2 h 9 min (APG Kern-Methode) und 13 min (SL Kern-Methode).

4.2.4 Vergleich der verschiedenen Modelle

Die in der 10-fachen-Kreuzvalidierung bestimmte Leistung des BioBERT-Modells übertrifft die beiden Kern-Methoden in allen bestimmten Parametern deutlich. So ist das bestimmte F-Maß um fast sechs Prozentpunkte höher als das der APG Kern-Methode. Das bedeutet, dass das BioBERT-Modell mithilfe seiner vielen Schichten aus Neuronen besser an den Trainingsdaten Muster und Abhängigkeiten generalisieren kann und dadurch eine bessere Leitung bei der Vorhersage erreicht. Der für die CPI-Klassifizierung ebenfalls aussagekräftige Parameter Genauigkeit ist um sieben Prozentpunkte höher. Das bedeutet, dass eine wirklich bestehende funktionale Interaktion eines CPI-Paars um sieben Prozent wahrscheinlicher richtig als funktional klassifiziert wird. Damit scheint das BioBERT-Modell prädestiniert für den Einsatz im RE.

Die Kehrseite der besseren Leistung von BioBERT ist die deutlich höhere benötigte Rechenleitung zum Trainieren der KNN. Dies äußert sich in der benötigten Laufzeit der Validierungen. Da für alle dieselben Daten verwendet wurden lassen sich die Werte direkt vergleichen. Extrem ist der Unterschied zwischen der SL Kern-Methode und BioBERT. So benötigt BioBERT für die Kreuzvalidierung mehr als das 53-fache der Laufzeit der SL Kern-Methode. Bei der APG Kern-Methode ist es ca. das 5-fache. Generell muss bedacht werden, dass für die Anwendung in der Literatur das Modell nur einmal trainiert werden muss. Daher ist es prinzipiell lohnend die dafür benötigte Zeit einmal zu investieren, um anschließend bessere Vorhersagen zu erzielen. Außerdem gibt es die Möglichkeit unter Verwendung von CUDA®, das Training über die GPU, anstelle der CPU laufen zu lassen. CUDA® ist eine von NVIDIA entwickelte Technologie und Programmierschnittstelle, welche es erlaubt die große Anzahl der Kerne der GPU für das Training von KNN zu nutzen. In einem allgemeinen Vergleich des Trainings über CPU und GPU aus der Literatur konnte die Trainingszeit um 85 % reduziert werden [43].

4.3 Einfluss der Trainingsdatensatzgröße

In diesem Kapitel werden die Ergebnisse der Untersuchungen mit der Veränderung der Trainingsdatensatzgröße dargestellt und diskutiert.

4.3.1 Einfluss auf die Leistung

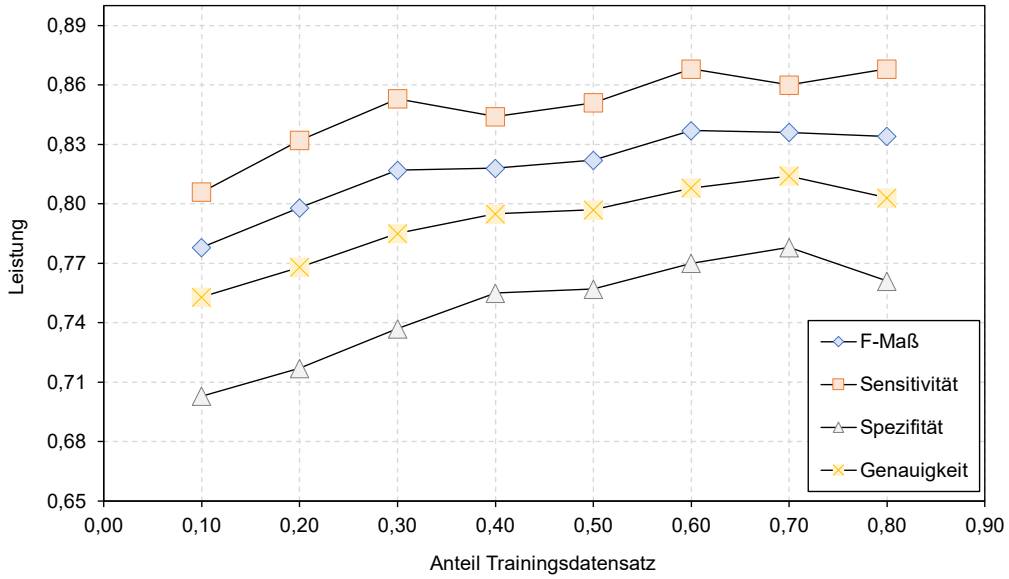

Abbildung 18: Zusammenhang von dem Anteil des Trainingsdatensatzes und der Leistung. Quelle: Eigene Darstellung

Die in Abbildung 18 dargestellten Werte stammen aus der Variation des Verhältnisses, indem der Datensatz in Test- und Trainingsdatensatz gesplittet wurde. Jeder Datenpunkt stammt aus dem arithmetischen Mittel von zehn zufällig im selben Verhältnis gesplitteten Datensätzen. Durch die Verwendung des Mittelwerts werden die durch mögliche Inhomogenitäten der Test- und Trainingsdatensätze verursachten Schwankungen gemittelt. Dem Graphen ist zu entnehmen, dass die Leistung des trainierten Modells bei den Vorhersagen am Testdatensatz mit zunehmend großen Testdatensatz zunimmt. Das gemittelte F-Maß bei einem Trainingsdatensatzanteil von 0,1 beträgt 0,778 und bei einem Trainingsdatensatzanteil von 0,8 beträgt es 0,834. Dabei ist der Anstieg des F-Maßes bei einem Trainingsdatensatzanteil von 0,1 auf 0,3 am größten. Dies ist damit zu begründen, dass der prozentuale Trainingsdatensatzgrößenzuwachs in diesem Bereich am höchsten ist. Mehr Trainingsdaten bedeuten eine größere Anzahl an Trainingsschritten, mit jeweils der Möglichkeit die Gewichte der Perzeptren so anzupassen, dass die Fehlerfunktion bis hin zu ihrem lokalen Minimum, immer kleiner wird. Negativ muss an diesem Versuch angemerkt werden, dass sich bei den verschiedenen Split-Verhältnissen die Größe des Testdatensatzes ebenfalls verändert. Damit kann der Einfluss der Trainingsdatensatzgröße nicht unabhängig vom

Testdatensatz betrachtet werden. Nichtsdestotrotz entspricht die beobachtete Abhängigkeit der Menge an Trainingsdaten von der Leistung des Modells den allgemeinen Erkenntnissen der Forschung im Bereich des maschinellen Lernens (siehe. Abbildung 19). Dabei wird differenziert zwischen dem klassischen maschinellen Lernen (z.B. die beiden Kern-Methoden) und tiefem Lernen (z.B. das BioBERT-Modell). Bei einer kleinen Trainingsdatenmenge erreichen Modelle des klassischen maschinellen Lernens eine bessere Leistung. Erst mit zunehmender Trainingsdatenmenge gelingt es auf tiefem Lernen basierenden Modellen besser zu generalisieren, sodass ab einer bestimmten Datenmenge die Leistung des klassischen maschinellen Lernens überschritten wird. Bei weiterer Vergrößerung der Menge an Trainingsdaten, stagniert die Leistung der Modelle des klassischen maschinellen Lernens, die Leistung von auf tiefem Lernen basierenden Modellen hingegen steigt weiter an [44].

Bei dem konkreten Datensatz aus dieser Arbeit steht fest, dass die Datensatzgröße den Punkt bereits erreicht hat, ab dem die Leistung der Modelle des tiefen Lernens (konkret BioBERT) die anderen Ansätze (Kern-Methoden) übersteigt (vgl. 4.2.4). Das resultiert in der Frage inwiefern eine Vergrößerung des Datensatzes Leistung des BioBERT-Modells erhöhen kann. Durch eine Betrachtung von Abbildung 18 lässt sich dies nicht klar beantworten. Es ist bei Trainingsdatensatzanteil von 0,6 bis 0,8 keine eindeutige Steigerung der Leistung des Modells zu erkennen.

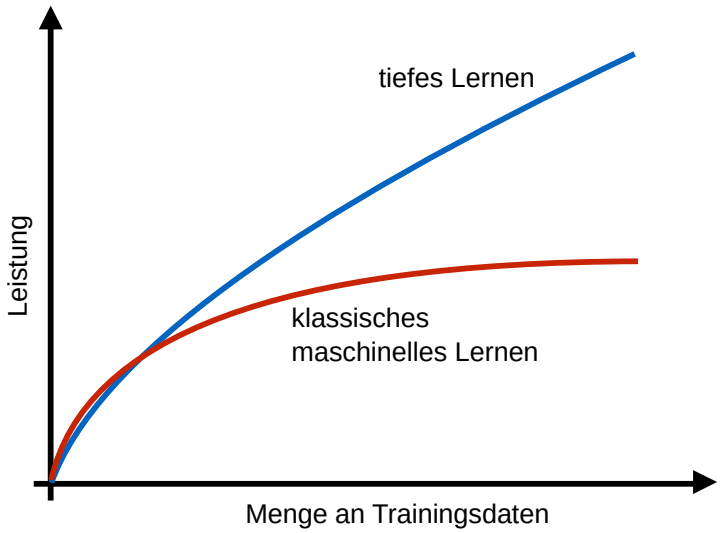

Abbildung 19: Abhängigkeit der Leistung eines Modells zu der Menge an Trainingsdaten. Quelle: Eigene Darstellung nach [45, S. 12]

4.3.2 Einfluss auf die Laufzeit

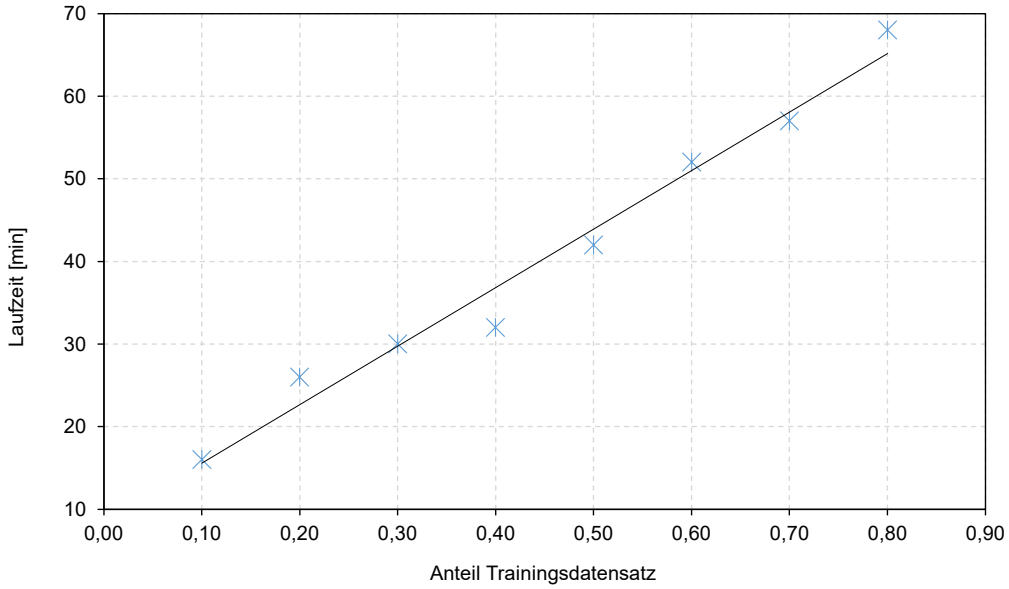

Abbildung 20: Zusammenhang des Anteils des Trainingsdatensatzes und der Laufzeit. Quelle: Eigene Darstellung

Als Nebeneffekt der Vergrößerung des Anteils des Trainingsdatensatzes erhöht sich die Laufzeit. Mit einem quadrierten Pearson-Korrelationskoeffizient von 0,978 kann der gefundene Zusammenhang als annähernd linear beschrieben werden. Da beim Training eines tiefen neuronalen Netzes alle Datenpunkte nacheinander zur Anpassung der Gewichte abgearbeitet werden, wirkt sich eine Vergrößerung der Anzahl der Daten direkt auf die Dauer des Trainings aus. Für jeden zusätzlichen Datenpunkt wird ein zusätzlicher Trainingsschritt benötigt. Das führt zu einer Zunahme der Trainingsdauer und damit zu einer Verlängerung der gesamten Laufzeit.

5 Abschlussbetrachtung

Im ersten Schritt wurde der Datensatz in die benötigte Form im TSV-Format überführt. In der Zukunft kann dieser Datensatz ebenfalls für andere Modelle, welche dieselbe Datenform erwarten, verwendet werden.

Anschließend konnte das BioBERT-Modell erfolgreich mit einer 10-fachen-Kreuzvalidierung evaluiert und mit den beiden Kern-Methoden verglichen werden. Als Ergebnis wurde herausgefunden, dass die Klassifizierungsleistung von BioBERT den beiden Kern-Methoden überlegen ist (F-Maß fast sechs Prozentpunkte höher). Um mit BioBERT ein komplett automatisches Text Mining in der Literatur durchzuführen, ist es notwendig zuerst die zu klassifizierenden Entitäten des CPI-Paars zu identifizieren. Dies ist eine NLP-Aufgabe, welche ebenfalls von BioBERT erledigt werden könnte (siehe *Named Entity Recognition*). Die Grundlage dafür bildet das Vortraining an ungelabelten Sätzen, das bereits in der Originalpublikation durchgeführt wurde.

Zu der letzten Teilaufgabe, dem Bestimmen des Einflusses der Größe der verwendeten Trainingsdatenmenge, kann abschließend gesagt werden, dass ein Zusammenhang von Leistung und Trainingsdatenmenge vor allem im Bereich einer geringen Datenmenge besteht. Die Frage, ob eine Vergrößerung des gesamten Datensatzes zu einer Steigerung der Klassifizierungsleistung des Modells führt, konnte nicht abschließend geklärt werden.

6 Ausblick

Die Bachelorarbeit hat gezeigt, dass BioBERT für die Verwendung im Bereich des CPI-Text Mining ein hohes Potential besitzt. Ein Schritt für die nahe Zukunft, könnte das Verwenden eines trainierten BioBERT-Modells zum RE aus der Gesamtzahl der PubMed Artikel sein. Aufgrund der riesigen Menge an Publikationen, ist eine längere Laufzeit zu erwarten. Daher ist es sinnvoll dies mit einem leistungsstärkeren Computer durchzuführen. Die aus der Vorhersage gewonnenen Informationen können mit den Ergebnissen der beiden Kern-Methoden in der Arbeit von *Döring K. et al.* verglichen werden [36]. Für die zukünftige Arbeit mit BioBERT ist es empfehlenswert das Finetuning des Modells über eine CUDA®-kompatible GPU laufen zu lassen, um Zeit einzusparen. Im Rahmen dieser Arbeit war das Testen von CUDA® bedauerlicherweise nicht möglich, da die vorhandene GPU nicht CUDA®-kompatibel ist.

Der Lehrstuhl Pharmazeutische Bioinformatik am Institut für Pharmazeutische Wissenschaften der Albert-Ludwigs-Universität Freiburg arbeitet aktuell unteranderem an einem CPI-Webserver. Dort sollen aus der Literatur extrahierte funktionale CPI-Paare zur Abfrage verfügbar sein. Für das Text Mining dieser funktionalen CPI-Paare wurden in der Publikation [36] bereits die beiden Kern-Methoden erfolgreich kombiniert. In der Zukunft könnte es ein Ziel sein, zusätzlich BioBERT in das Klassifizierungssystem zu integrieren. Ein einfacher Ansatz wäre es ein CPI-Paar nur abschließend als funktional zu bewerten, wenn alle drei Modelle eine funktionale Interaktion vorausgesagt haben. Es ist zu erwarten, dass auf diesem Weg eine höhere Genauigkeit bei der Klassifizierung erreicht werden kann.

Ein weiterer wichtiger Ansatzpunkt zur Optimierung stellt der in der Arbeitsgruppe manuell erzeugte Datensatz dar. So könnte, wenn er weiter vergrößert wird, untersucht werden, wie stark die Leistung des Modells verbessert werden kann. Bei einer Vergrößerung könnte z.B. der Fokus auf das Hinzufügen von komplexeren CPI-Paaren gesetzt werden. Denn, wenn dem Modell für das Training mehr solcher Randfälle zur Verfügung stehen, ist zu erwarten, dass das Modell bei unbekannten Randfällen besser klassifizieren kann. Das manuelle Vergrößern des Datensatzes stellt eine zeit- und damit auch geldaufwendige Aufgabe dar. Möglichkeiten die Arbeit zu erleichtern, könnten entweder die Zuhilfenahme von semi-manuellen Methoden oder die gezielte Integration eines Teils des BioCreativ VI Datensatzes sein. Generell ist es mit Spannung zu erwarten, welche neuen auf tiefes Lernen basierende Modelle für das RE in den nächsten Jahren publiziert werden. Es scheint nur eine Frage der Zeit, bis das BioBERT-Modell in der Leistung übertroffen werden wird.

7 Literaturverzeichnis

[1] D. Mendez *et al.*, "ChEMBL: Towards direct deposition of bioassay data," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D930–D940, 2019.

[2] 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.

[3] 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.

[4] "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).

[5] 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.

[6] 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.

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

[8] 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.

[9] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[10] "Wie KI der Materialforschung nutzt." https://www.selfmem.eu/public_relations_media/magazine/what_motivates_us/081840/index.php.de (accessed Mar. 02, 2021).

[11] 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 (accessed Mar. 02, 2021).

[12] R. Fjelland, "Why general artificial intelligence will not be realized," *Humanit. Soc. Sci. Commun.*, vol. 7, no. 1, pp. 1–9, 2020.

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

[14] S. Badillo *et al.*, "An Introduction to Machine Learning," *Clin. Pharmacol. Ther.*, vol. 107, no. 4, pp. 871–885, 2020.

[15] “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).

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

[17] “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).

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

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

[20] 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.

[21] 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.

[22] A. Scherer and A. Scherer, “Das Perzeptron,” *Neuronale Netze*, pp. 65–70, 1997.

[23] “Neuronale Netze; Univerität Ulm.” http://www.mathematik.uni-ulm.de/stochastik/lehre/ss07/seminar_sl/ausarbeitung_wallner.pdf (accessed Feb. 27, 2021).

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

[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.

[29] F. Rinaldi *et al.*, “Open Access OntoGene in BioCreative II,” vol. 9, no. Suppl 2, pp. 1–11, 2008.

[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.

[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.

[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.

[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.

[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.

[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.

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

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

[38] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[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.

[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.

[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.

[43] “TensorFlow 2 - CPU vs GPU Performance Comparison.” https://datamadness.github.io/TensorFlow2-CPU-vs-GPU (accessed Mar. 21, 2021).

[44] M. Z. Alom *et al.*, “A state-of-the-art survey on deep learning theory and architectures,” *Electron.*, vol. 8, no. 3, pp. 1–67, 2019.

[45] A. Zappone, M. Di Renzo, and M. Debbah, “Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both?,” *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, 2019.