Ostfalia
Hochschule für angewandte Wissenschaften
Wolfenbüttel

Fakultät Informatik

Erkan Garan, 70467533

Entwurf eines Modells für ein unterstützendes KI-System zur Bewertung von Anwendungsregeln

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Science im Studiengang Informatik im Praxisverbund

an der Ostfalia Hochschule

Hochschule Braunschweig/Wolfenbüttel

Betreuer:
Prof. Dr. Claus Fühner
Dipl. -Inf. Stefan Jung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist.

Wolfenbüttel, den 8. März 2023

Kurzfassung

Eine besondere Form von Anforderungen stellen die Anwendungsregeln dar. Um einen sicheren und zuverlässigen Einsatz von Komponenten in einem Projekt zu gewährleisten, ist eine Erfüllung der Anwendungsregeln der im Projekt genutzten Komponenten vorausgesetzt. Diese Bachelorarbeit wird sich mit der Erstellung eines unterstützenden KI-Systems beschäftigen, das in der Lage dazu sein soll, mögliche Vorschläge für Bewertungen zu Anwendungsregeln zu liefern. Dafür wird ein Deep Learning Modell mit Daten über Projekte und ihren Anwendungsregeln der Siemens Mobility GmbH trainiert. Diese Daten müssen vorher in eine für das Anlernen eines Modells geeignete Form gebracht werden. Diese Schritte der Datenvorverarbeitung werden hier vorgestellt. Zudem werden in dieser Arbeit Kenntnisse darüber vermittelt, wie künstliche Intelligenz funktioniert, was der Unterschied zwischen KI, maschinellem Lernen und Deep Learning ist und wie solch ein Modell aufgebaut, angewendet und getestet werden kann.

Abstract

Application rules are a special type of requirements. To ensure the safe and reliable use of components in a project, a fulfillment of the application rules from the components used in the project is mandatory. This bachelor thesis will deal with the creation of a supporting AI system, that should be able to predict possible suggestions for the evaluation of application rules. For this purpose, a deep learning model will be trained with data on projects and their application rules from the Siemens Mobility GmbH. This data must first be transformed into a form suitable for the training of an AI model. The required data preprocessing steps are presented here. In addition, this thesis will provide knowledge about, how artificial intelligence works, what the difference between AI, machine learning and deep learning is and how such a model can be built, applied and tested.

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

DOORS IBM Rational DOORS

DXL DOORS eXtension Language

KI Künstliche Intelligenz

ML Machine Learning

DL Deep Learning

NN Neuronales Netz

ReLU Rectified Linear Unit

RE Requirements Engineering

RM Requirements Management

IREB International Requirements Engineering Boards

SMO RI Siemens Mobility Rail Infrastructure

1 Einleitung

1.1 Motivation

1.2 Zielsetzung

1.3 Aufbau

2 Grundlagen

Die Themenbereiche Requirements Engineering (RE), speziell das Bewerten von Anwendungsregeln, und Künstliche Intelligenz (KI) stellen den Schwerpunkt dieser Bachelorarbeit dar und werden in diesem Kapitel grundlegend vorgestellt. Zudem wird das Anforderungsmanagement-Tool IBM Rational DOORS (DOORS) vorgestellt, da es sowohl von der Siemens Mobility GmbH genutzt wird als auch in dieser Arbeit Verwendung finden wird.

2.1 Requirements Engineering

Nach der Definition des International Requirements Engineering Boards (IREB) bezeichnet das RE die systematische und disziplinierte Vorgehensweise bei der Spezifikation und dem Management von Anforderungen. Das Ziel des RE ist dabei, die Wünsche und Bedürfnisse der Stakeholder zu verstehen [3, vgl. S.30]. Stakeholder sind Personen oder Organisationen, die die Anforderungen des Systems direkt oder indirekt beeinflussen oder die von dem System betroffen sind [3, vgl. S.33]. Beispielsweise können Kunden oder Nutzer, aber auch der Gesetzgeber, potenzielle Stakeholder sein. Außerdem soll das Risiko minimiert werden, dass diese Wünsche und Bedürfnisse nicht oder nur unzureichend erfüllt werden [3, vgl. S.30].

Einen Teilbereich des RE stellt das Requirements Management (RM) dar. Dieser Prozess beschreibt die Verwaltung, Speicherung, Änderung sowie die Rückverfolgung von Anforderungen [3, vgl. S.8]. Um den RM Prozess zu unterstützen, können entsprechende Tools verwendet werden. Bei der Siemens Mobility GmbH wird das Tool DOORS verpflichtend eingesetzt [4, vgl. S.18]. Näheres zu DOORS kann dem Kapitel 2.2 entnommen werden.

2.1.1 Anforderungen

Die IEEE definiert eine Anforderung wie folgt:

> „(1) A condition or capability needed by a user to solve a problem or achieve an objective.
> (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
> (3) A documented representation of a condition or capability as in (1) or (2).“ [5, S.62]

Daher bilden Anforderung die Basis eines jeden Projekts, da diese definieren, welche Bedingungen ein System erfüllen muss bzw. welche Fähigkeiten es besitzen muss. Sie werden idealerweise unter Berücksichtigung und in Zusammenarbeit mit den Stakeholdern des Projekts ermittelt. Neben den Stakeholdern können unter anderem auch Normen, Gesetze oder Vorgänger eines Systems weitere Quellen für Anforderungen sein. Um von jedem Partizipierenden des Projekts verstanden zu werden, werden Anforderungen in der Regel in natürlicher Sprache formuliert. Da natürliche Sprache Raum für Interpretation bieten kann, muss darauf geachtet werden, dass die Anforderungen so klar und unmissverständlich wie möglich formuliert werden und dabei trotzdem vollständig bleiben. Zudem wird voraussichtlich nicht jeder Stakeholder über die fachlichen Kenntnisse verfügen um Fachsprache oder Konventionen zu verstehen, weshalb darauf verzichtet werden sollte [6, vgl. S.2]. Um sicherzustellen, dass Anforderungen korrekt formuliert werden, wurden in der Norm ISO/IEC/IEEE 29148:2018(E) Eigenschaften definiert, welche Anforderungen erfüllen sollen. Diese Eigenschaften werden nachfolgend dargestellt und kurz beschrieben:

Notwendig Die Anforderung definiert eine wesentliche Fähigkeit, Eigenschaft, Einschränkung und/oder einen Qualitätsfaktor [7, vgl. S.12].

Angemessen Die Anforderung verfügt über einen angemessenen Detaillierungsgrad und erlaubt dabei bei der Implementierung größtmögliche Unabhängigkeit [7, vgl. S.12].

Eindeutig Die Anforderung ist leicht zu verstehen, einfach formuliert und kann nur auf eine einzige Weise interpretiert werden [7, vgl. S.12].

Komplett Die Anforderung ist hinreichend beschrieben und benötigt keine weiteren Informationen um verstanden zu werden [7, vgl. S.12].

Atomar Die Anforderung beschreibt eine einzige Fähigkeit oder Bedingung [7, vgl. S.12].

Durchführbar Die Anforderung kann innerhalb der Beschränkungen des Systems mit akzeptablem Risiko durchgeführt werden [7, vgl. S.13].

Verifizierbar Die Umsetzung der Anforderung kann überprüft werden [7, vgl. S.13].

Korrekt Die Anforderung ist eine genaue Darstellung des Bedürfnisses ihrer Quelle [7, vgl. S.13].

Konform Die Anforderung wurde, wenn möglich, mithilfe einer genehmigten Standardvorlage und -stil verfasst [7, vgl. S.13].

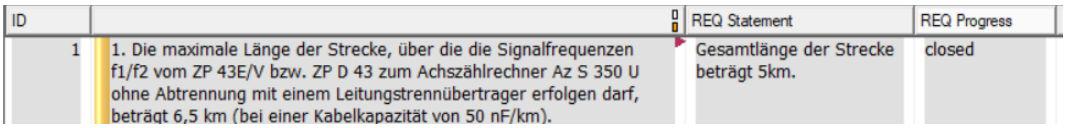
Um diese Eigenschaften zu erfüllen, ist es ratsam auf Vorlagen für das Schreiben von Anforderungen zurückzugreifen. Eine Anforderung, welche mithilfe einer Vorlage verfasst wurde, könnte wie folgt aussehen:

> „The <system> shall <function> <object> every <performance> <units>.
> E.g. The coffee machine shall produce a hot drink every 10 seconds. “
> [6, S.81]

Nach einer Umfrage aus dem Chaos Report der Standish Group nennen mehr als die Hälfte der Befragten als Faktor für die Beeinträchtigung von Projekten einen Grund, der direkt im Zusammenhang mit mangelndem RE und RM steht. Dazu gehören z.B. Gründe wie unvollständige Anforderungen, Nutzer nicht ausreichend involviert, unrealistische Erwartungen oder geänderte Anforderungen und Spezifikationen [8, vgl. S.5]. Gut durchgeführtes RE und RM ist also essenziell für den Erfolg von Projekten.

2.1.2 Anwendungsregeln

Eine spezielle Form von Anforderungen stellen die Anwendungsregeln dar. Diese Anforderungen werden an Komponenten gestellt, um einen sicheren und zuverlässigen Einsatz von Komponenten im System zu gewährleisten. Dafür müssen sie von dem Kunden oder dem Projekt, welches die jeweiligen Komponenten nutzt, berücksichtigt werden [9, vgl. S.9]. Alle Anwendungsregeln von Komponenten der Siemens Mobility Rail Infrastructure (SMO RI) in all ihrer Versionen befinden sich zentral gespeichert in einem Projekt im Tool DOORS. Von dort kann der zuständige System-Manager eines Projekts alle Anwendungsregeln von Komponenten importieren, die im jeweiligen Projekt genutzt werden. Als Nächstes muss er bewerten, welche Anwendungsregeln für das Projekt anwendbar sind und welche nicht. Die anwendbaren Anwendungsregeln müssen daraufhin mit einer Designlösung abgeschlossen werden oder einem oder mehreren Teilsystemen zugeordnet werden. Diese Bewertung der Anwendungsregeln erfolgt ebenfalls in DOORS wie in der Abbildung 2.1 gezeigt wird.



ID		REQ Statement	REQ Progress
1	1. Die maximale Länge der Strecke, über die die Signalfrequenzen f1/f2 vom ZP 43E/V bzw. ZP D 43 zum Achszählrechner Az S 350 U ohne Abtrennung mit einem Leitungstrennübertrager erfolgen darf, beträgt 6,5 km (bei einer Kabelkapazität von 50 nF/km).	Gesamtlänge der Strecke beträgt 5km.	closed

Abbildung 2.1: Bewerten einer Anwendungsregel

Dafür werden die Attribute REQ Statement und REQ Progress genutzt. Im Attribut REQ Progress kann der System-Manager eine von fünf verschiedenen Ausprägungen wählen. Eine Auflistung und Erklärung dieser Ausprägungen kann der Tabelle 2.1 entnommen werden. Der System-Manager muss daraufhin ein Statement darüber abgeben, warum er welchen REQ Progress gewählt hat. Dieses Statement wird in Textform im Attribut REQ Statement gespeichert. Die Anwendungsregel in der Abbildung 2.1 wurde beispielsweise mit closed bewertet, da die Anwendungsregel für das Projekt anwendbar ist und zudem auch abgeschlossen ist, da die Streckenlänge, wie in der Anwendungsregel gefordert, weniger als 6,5 km beträgt.

Tabelle 2.1: Ausprägungen des Attributs REQ Progress [9, vgl. S.28f.]

REQ Progress	Erklärung
not applicable	Anwendungsregel ist nicht anwendbar
forwarded	Anwendungsregel soll an Subsystem weitergeleitet werden
closed	Anwendungsregel auf System Design Ebene mit einem Statement abgeschlossen
open	Bewertung der Anwendungsregel noch offen
compliant	Anwendungsregel anwendbar, noch nicht bewertet

2.2 IBM Rational DOORS

Das Anforderungsmanagement-Tool DOORS ist ein plattformübergreifendes und unternehmensweites Tool und wird zur Erfassung, Verknüpfung, Verfolgung, Analyse und Verwaltung von Anforderungen genutzt. DOORS ist ein Akronym, das für Dynamic Object-Oriented Requirements System steht. Alle Anforderungen und weitere Informationen werden in einer zentralen Datenbank gespeichert. Innerhalb der Datenbank werden die Informationen in Modulen gespeichert. Diese Module können mithilfe von Ordnern und Projekten organisiert werden. Ordner sind vergleichbar mit den Ordnern z.B. im Windows Explorer und können andere Ordner, Projekte oder Module beinhalten. Ein Projekt hingegen ist ein spezieller Ordner, der alle Daten für ein entsprechendes Projekt beinhaltet. Sowohl für Ordner als auch für Projekte können die Zugriffsrechte individuell eingestellt werden [6, vgl. S.173]. Dabei existieren die Optionen read, modify, create, delete und administer (RMCDA).

2.2.1 Module

Es existieren zwei verschiedene Arten von Modulen im Anforderungsmanagement-Tool DOORS. Module, die die eigentlichen Anforderungen beinhalten, werden Formal Module genannt. Abbildung 2.2 zeigt ein Beispiel für so ein Modul. Zu erkennen ist dort ein geöffnetes Formal Module. Auf der linken Seite ist ein Explorer

zu sehen, auf der rechten Seite die eigentlichen Inhalte des Moduls. Durch den Explorer auf der linken Seite, der in einer Baumstruktur organisiert ist, wird es dem User ermöglicht leicht zu einer bestimmten Stelle im Modul zu navigieren. Dabei können die einzelnen Sektionen auf- und zugeklappt werden [6, vgl. S.176]. Die Daten auf der rechten Seite sind tabellarisch angeordnet. Die Spalten stellen dabei die einzelnen Attribute des Moduls dar, während die Zeilen die Objekte darstellen.

Neben den Formal Modules existieren auch die Link Modules. In diesen werden Informationen über die Beziehungen zwischen einzelnen Objekten gespeichert.

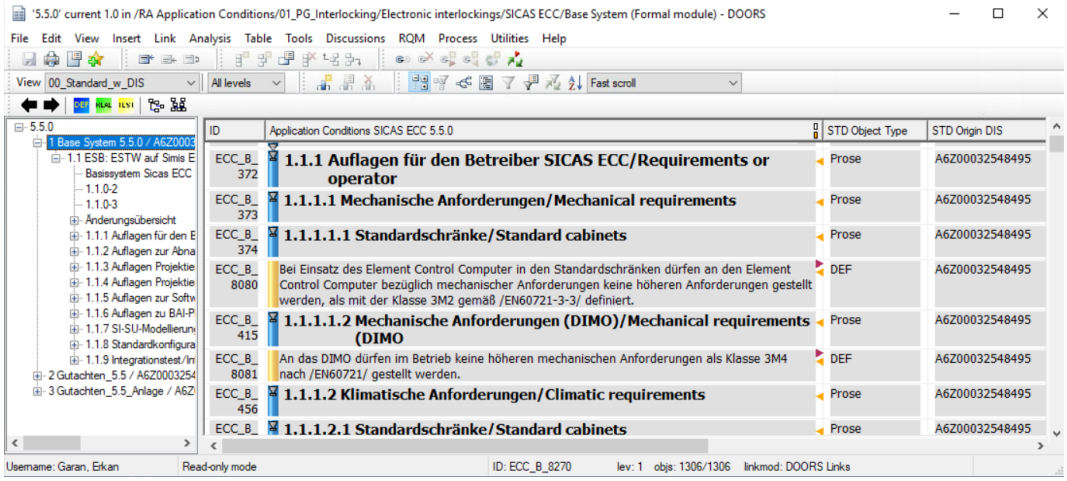


Abbildung 2.2: Geöffnetes Modul in DOORS

2.2.2 Objekte und Attribute

Innerhalb eines Moduls werden Daten in Objekten gespeichert. In der Regel bestehen Objekte aus mindestens zwei Spalten. Die erste Spalte enthält eine ID, die sich aus einem Präfix und einem Integerwert zusammensetzt. Der Integerwert wird bei jedem neu angelegtem Objekt inkrementiert, sodass jedem Objekt innerhalb eines Moduls eine eindeutige ID zugeordnet werden kann. Die zweite Spalte besteht dabei entweder aus einer Sektions-Nummer und einer Überschrift, wie im ersten Objekt in der Abbildung 2.2 zu sehen ist, oder aus einem Objekt-Text, der beispielsweise eine Anforderung beinhalten kann. Ein Beispiel für einen Objekt-Text, der eine Anforderung beinhaltet, ist das vierte Objekt der Abbildung 2.2 [6, vgl. S.178]. Einem Objekt können beliebig viele weitere Attribute hinzugefügt werden.

Attribute beinhalten relevante Informationen über Module oder Objekte. Modulattribute speichern Informationen über das Modul, wie beispielsweise den Ersteller des Moduls, das letzte Änderungsdatum und Ähnliches. Diese Modulattribute findet der User über die Eigenschaften des Moduls, welche mit einem Rechtsklick auf das Modul in der grafischen Oberfläche geöffnet werden können. Objektattribute hingegen speichern Informationen über die Objekte. In der Abbildung 2.2 ist die Spalte STD Object Type z.B. ein Objektattribut, das definiert, ob es sich bei dem Objekt um eine Anforderung oder um Prosa, also z.B. eine Überschrift handelt.

2.2.3 Baseline

Eine Baseline friert den aktuellen Stand der Anforderungen eines Projekts mit ihren Attributen ein und ist eine nicht veränderbare Kopie von formalen Modulen [6, vgl. S.182]. Baselines werden in der Regel zu Releases von Systemen oder Subsystemen erstellt [4, vgl. S.60]. Wenn ein Formal Module geöffnet ist, kann der User unter File → Baseline eine Baseline erstellen oder eine bereits vorhandene Baseline ansehen. Abbildung 2.3 zeigt das Dialogfenster zum Öffnen bereits vorhandener Baselines. Dort wird deutlich, dass Baselines versioniert werden können.

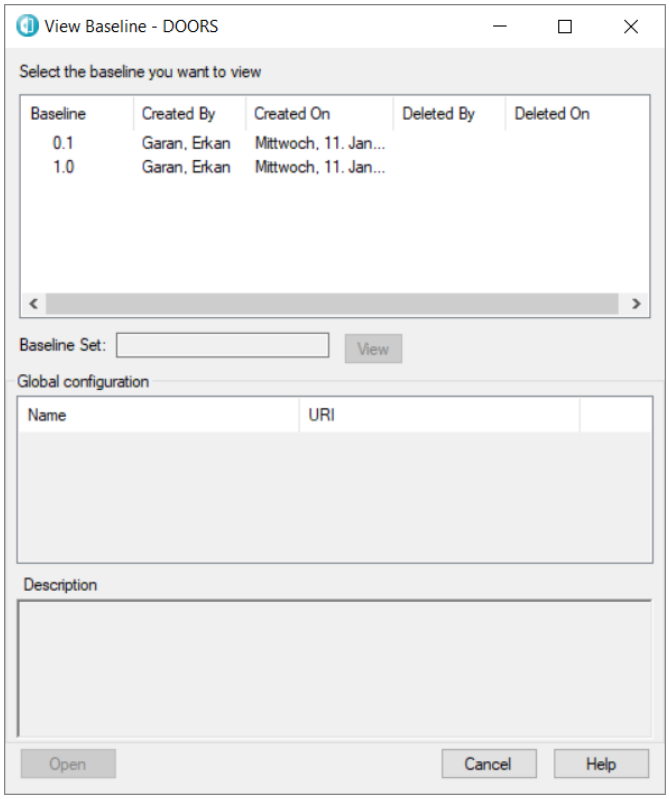


Abbildung 2.3: Baselines eines Moduls in DOORS

2.2.4 Links

In der Abbildung 2.2 sind an der rechten Kante der zweiten Spalte zwei Arten von Pfeilen zu erkennen. Diese Pfeile symbolisieren Links in DOORS. Links sind gerichtete Verbindungen von einem Quellobjekt zu einem Zielobjekt. Sie werden in DOORS genutzt, um die Verfolgbarkeit von Anforderungen zu gewährleisten. Der User kann dabei, ungeachtet von der Richtung des Links, vom Quellobjekt zum Zielobjekt oder andersherum navigieren [6, vgl. S.183]. Ein nach links zeigender gelber Pfeil ist dabei ein In-Link, das heißt, dass dieses Objekt als Zielobjekt dient und ein anderes Objekt eine Verbindung zu diesem Objekt hat. Das Gegenstück zum In-Link ist ein Out-Link. Dieser wird in der grafischen Benutzeroberfläche von DOORS als roter nach rechts zeigender Pfeil dargestellt. Hat ein Objekt einen Out-Link, heißt das, dass dieses Objekt als Quellobjekt dient und eine Verbindung zu einem anderen Objekt, welches als Zielobjekt dient, hat.

2.2.5 DXL

DOORS eXtension Language (DXL) ist eine Skript-Sprache, die speziell für das Anforderungsmanagement-Tool DOORS entwickelt wurde. Durch diese Skript-Sprache besteht die Möglichkeit, die grafische Benutzeroberfläche von DOORS um neue, entwickelte Anwendungen zu erweitern. Von der Syntax ähnelt die Sprache den Programmiersprachen C und C++ [10, vgl. S.1]. Zudem können Skripte geschrieben werden, die als Batch-Skript ausgeführt werden können. Diese Skripte bieten neben der grafischen Benutzeroberfläche eine weitere Möglichkeit, um mit DOORS zu arbeiten. DXL wurde im Praxisprojekt zum Sammeln von bewerteten Anwendungsregeln aus Projekten der Siemens Mobility GmbH genutzt, indem Batch-Skripte geschrieben wurden, welche im zentralen Projekt, das die Anwendungsregeln von Komponenten beinhaltet, die die Links der Objekte verfolgt haben und dort in den Modulen nach korrekt bewerteten Anwendungsregeln gesucht haben. Diese wurden dann nach Projekt und Komponente gruppiert und jeweils in neue Module geschrieben. Der Inhalt dieser Module dient als Datensatz für diese Bachelorarbeit. Ebenfalls wird DOORS dazu benötigt, um in der grafischen Oberfläche eines Moduls mit neu importierten Anwendungsregeln ein Programm zu starten, was mit den Informationen aus dem Modul ein weiteres Skript in der Programmiersprache Python startet, in der das KI-Modell erstellt und angelernt wird und mit den Daten über die Anwendungsregeln aus dem Modul Vorschläge zur Bewertung der Anwendungsregeln liefern soll.

2.3 Künstliche Intelligenz

Das Thema KI hat in letzter Zeit durch den Chatbot ChatGPT der Firma OpenAI eine große mediale Aufmerksamkeit erhalten. ChatGPT wurde im November 2022 veröffentlicht und ist ein Sprachmodell, das natürliche Sprache verstehen und abhängig von der Benutzereingabe in Dialogform Antworten liefern soll. Es ist dabei in der Lage, sich an vorherige Eingaben zu erinnern und kann deshalb Anfragen in einen Kontext einordnen. Dadurch kann es auch vorherige Antworten korrigieren und auf Wünsche und Bedürfnisse des Benutzers eingehen [11].

Die Frage nach KI und nach dem ob und wie Maschinen denken können ist jedoch wesentlich älter als dieser Chatbot. Bereits Alan Turing stellte sich diese Frage in den 1950er Jahren. Er schlug vor, die Frage, ob Maschinen denken können, mit einer anderen Frage zu ersetzen. Dafür formulierte er das Problem mithilfe eines Spiels, dem „Imitation Game“, das heute eher als Turing-Test bekannt ist, um. Dieses Spiel wird mit drei Personen gespielt, einem Fragesteller, einem Mann und einer Frau. Ziel des Spiels ist es, dass der Fragesteller mithilfe von Fragen herausfinden soll, welche Person der Mann und welche Person die Frau ist. Der Mann soll dabei versuchen, den Fragesteller dazu zu bringen, ihn fälschlicherweise als die Frau zu identifizieren, während die Frau versuchen soll, dass der Fragesteller sie korrekt als Frau identifiziert. Dafür befindet sich der Fragesteller in einem anderen Raum und die Fragen werden entweder über eine außenstehende weitere Person oder einem Fernschreiber übermittelt. Was würde nun passieren, wenn eine Maschine die Rolle des Mannes übernehmen würde? Würde der Fragesteller dann häufiger oder seltener gewinnen? Alan Turing glaubte, dass eine Maschine dann als intelligent bezeichnet werden könne, wenn eine Maschine in der Lage dazu wäre, menschliches Verhalten zu imitieren [12, vgl. S.433f.].

François Chollet, der Entwickler der Deep-Learning-Bibliothek Keras, welche im Kapitel TODO näher beschrieben wird, definiert das Fachgebiet KI als „[den] Versuch, normalerweise von Menschen erledigte geistige Aufgaben automatisiert zu lösen“ [1, S.22]. Nach dieser Definition schließt KI weitere Themen wie das Machine Learning (ML) sowie das Deep Learning (DL) ein. Wichtig zu beachten ist dabei, dass diese Gebiete sich voneinander unterschieden. Ihre Beziehung zueinander wird in Abbildung 2.4 verdeutlicht.

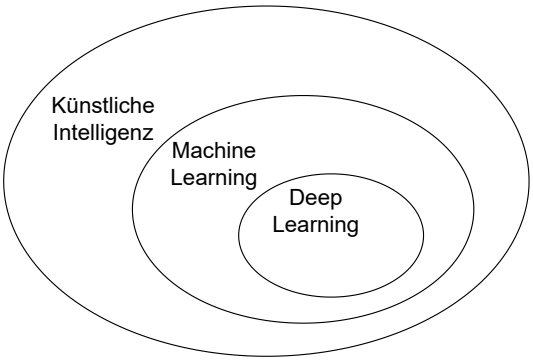


Abbildung 2.4: Beziehung zwischen KI, ML und DL [1, S.22]

Dieses Kapitel wird näher erläutern, worum es sich bei den Themen ML und DL handelt und wie ein KI-Modell als neuronales Netz erstellt wird.

2.3.1 Maschinelles Lernen

Das maschinelle Lernen ist ein Teilgebiet der KI und beschäftigt sich mit der Frage, ob ein Computer in der Lage dazu ist selbstständig eine bestimmte Aufgabe zu erlernen. Ziel dabei ist es, dass eine Maschine aus einem vorgegebenem Datensatz und den dazugehörigen Antworten Regeln extrahieren soll, die den Datensatz erklären können. Anders als bei der klassischen Programmierung soll eine Maschine hier nicht die Antworten aus den Daten und Regeln herausgeben, sondern selber nach einer Struktur suchen, aus der die Maschine dann Regeln ableiten kann, die auch auf andere Aufgaben angewendet werden können [1, vgl. S.23f.]. Angenommen ein Datensatz bestünde aus Bildern von Hunden und Menschen. Bei der klassischen Programmierung würde der Programmierer nun selber Regeln definieren und diese programmieren müssen. Beispiele für mögliche Regeln könnten hier sein:

> Wenn Wesen Fell hat, dann ist es ein Hund.
> Wenn Wesen auf zwei Beinen läuft, dann ist es ein Mensch.

Beim ML hingegen müsste der Programmierer diese Regeln nicht selber definieren. Hier werden neben dem Datensatz noch die entsprechenden Antworten benötigt. Jedes Bild bräuchte also ein Label, also eine Information darüber, ob es sich auf dem Bild um einen Hund oder einen Menschen handelt. Aufgabe der Maschine wäre es nun, mithilfe des Datensatzes und der Label, Regeln zu definieren, ob auf einem Bild ein Hund oder ein Mensch zu sehen ist. Das ML-System wird also sozusagen trainiert. Dem trainiertem System können dann neue Bilder gezeigt werden und es wäre in der Lage anhand seiner definierten Regeln vorherzusagen, ob auf den neuen Bildern Hunde oder Menschen zu sehen sind.

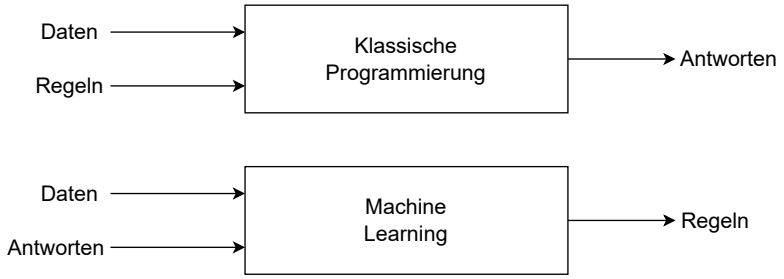


Abbildung 2.5: Unterschiedliche Programmierparadigmen [1, S.23]

Abbildung 2.5 verdeutlicht nochmal die Unterschiede zwischen der klassischen Programmierung und dem ML. Beim ML sind also drei Elemente notwendig, diese werden anhand des oben genannten Beispiels verdeutlicht:

Eingabedaten Bilder von Menschen und Hunden [1, S.24]

Antworten Label, ob auf dem Bild ein Mensch oder ein Hund abgebildet ist [1, S.24]

Metrik zur Bewertung des Algorithmus Benötigt, um die Abweichung zwischen Ausgabe des Modells und eigentlicher Antwort zu bestimmen. Wird als Feedback-Signal genutzt, um Algorithmus anzupassen. Beispielsweise wie viel Prozent der Bilder richtig zugeordnet werden. [1, S.24f.].

Die Aufgabe eines ML-Modells ist es passende Repräsentationen der Eingabedaten zu erlernen, das heißt, dass die Daten vom Modell sinnvoll umgewandelt werden müssen. Die systematische und automatische Suche nach der bestmöglichen Repräsentationen der Daten, mithilfe eines Feedback-Signals, für eine bestimmte vorgegebene Aufgabe kann als Möglichkeit verstanden werden, wie Maschinen lernen können. Mithilfe dieses Ansatzes und der in den letzten Jahren besser werdenden Hardware können große Datenmengen analysiert werden, was das Lösen von Aufgaben wie der Spracherkennung oder dem autonomen Fahren ermöglicht hat [1, vgl. S.24ff.].

Die hier beschriebene Variante des Machine Learnings wird auch Supervised Learning, also überwachtes Lernen, genannt. „Supervised“ bezieht sich hierbei darauf, dass dem Modell die Antworten vorgegeben werden und es anhand der Antworten versucht Regeln zu finden. Neben dieser Variante existiert auch noch das sogenannte Unsupervised Learning. Bei dieser Variante des Machine Learnings werden dem Modell keine Antworten gegeben. Stattdessen versucht das Modell die Daten anhand ihrer Beziehung zueinander zu analysieren und die Daten in Kategorien zu klassifizieren. Algorithmen zum Clustern von Datensätze sind typische Beispiele für das Unsupervised Learning [13, vgl. S.47ff.]. In dieser Arbeit finden Unsupervised Learning Algorithmen oder Modelle keine Anwendung, deshalb ist mit ML hier immer Supervised Learning gemeint.

2.3.2 Deep Learning und neuronale Netze

Wie Abbildung 2.4 zeigt, ist das DL ein Teilgebiet des ML und beschäftigt sich somit ebenfalls mit der Suche nach den bestmöglichen Repräsentationen von Daten. Das Besondere am DL ist, dass das Lernen des Modells in mehreren aufeinanderfolgenden Schichten, auch Layer genannt, stattfindet. Dabei können beliebig viele Schichten eingesetzt werden. Die Anzahl an Schichten wird auch Tiefe des Modells genannt, was auch das „Deep“ in DL beschreibt. Je tiefer dabei eine Schicht liegt, desto sinnvoller sollen die Repräsentationen werden [1, vgl. S.27]. François Chollet definiert DL als „Lernen durch schichtweise Repräsentationen oder Lernen durch hierarchische Repräsentationen“ [1, S.27]. DL-Modelle werden in der Regel als Neuronales Netz (NN) dargestellt, das aus beliebig vielen Schichten besteht, wobei jedes NN mindestens eine Eingabe- und eine Ausgabeschicht besitzt, dazwischen kann es beliebig viele versteckte Schichten, die hidden Layer, beinhalten. Die versteckten Schichten sind für die Verarbeitung der Daten zuständig. Eine Schicht besteht dabei aus einer vom Ersteller des NN ausgewählten Anzahl an Neuronen, die miteinander verbunden sind [2, vgl. S.26]. In der Abbildung 2.6 wird der grundlegende Aufbau eines einfachen neuronalen Netzes dargestellt.

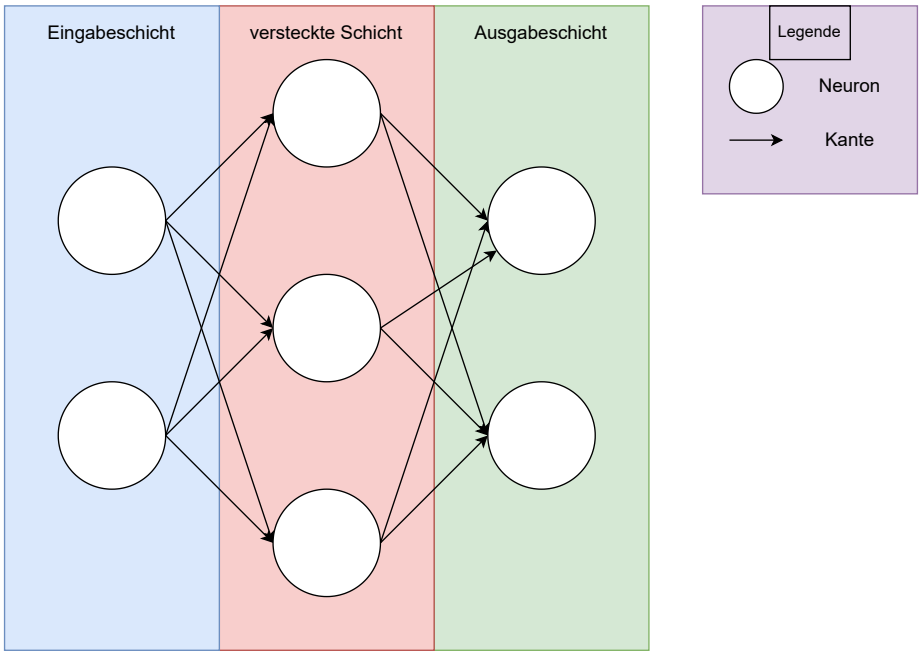


Abbildung 2.6: Aufbau eines neuronalen Netzes [2, S.27]

Die Verbindungen zwischen den Neuronen, die Kanten genannt werden, sind gerichtet und besitzen ein Gewicht. Wenn jedes Neuron eine Verbindung zu jedem Neuron in der nächsten Schicht hat, dann wird diese Schicht auch als fully-connected bezeichnet. Das NN in Abbildung 2.6 besteht aus drei Schichten, welche, bis auf die Ausgabeschicht, fully-connected sind. Unabhängig davon bekommt jedes Neuron, welches sich nicht in der Eingabeschicht befindet, eine Netzeingabe, welche sich aus den Ausgaben der Neuronen zusammensetzt, die eine Kante zu dem jeweiligen Neuron haben. Dabei wird jede Ausgabe eines Neurons mit dem Kantengewicht multipliziert.

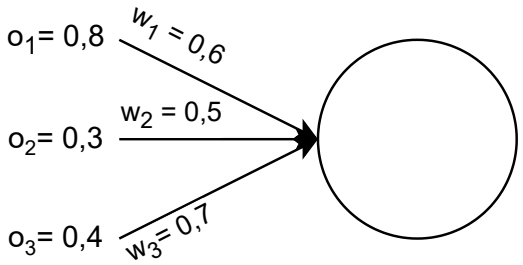


Abbildung 2.7: Berechnung Netzeingabe [2, vgl. S.29]

Abbildung 2.7 zeigt beispielhaft ein Neuron und die Ausgabe von drei vorherigen Neuronen. Die Netzeingabe des Neurons wird wie folgt berechnet [2, S.29]:

$$net = o_1 \cdot w_1 + o_2 \cdot w_2 + o_3 \cdot w_3 = 0,8 \cdot 0,6 + 0,3 \cdot 0,5 + 0,4 \cdot 0,7 = 0,91 \quad (2.1)$$

Jedes Neuron besitzt eine Aktivierungsfunktion, welche mithilfe der Netzeingabe prüft, ob das jeweilige Neuron eine Ausgabe hat und welchen Wert diese annimmt. Es existieren viele verschiedene Aktivierungsfunktionen, in dieser Arbeit wird die Rectified Linear Unit (ReLU)-Funktion genutzt. Diese ist definiert als [2, S.31]:

$$f(net) = max(0; net) \quad (2.2)$$

Diese Funktion nimmt die Netzeingabe und wenn die Netzeingabe positiv ist, dann wird die Netzeingabe als Ausgabe ausgegeben, ansonsten gibt das Neuron 0 als Ausgabe aus. Eine ReLU-Funktion als Aktivierungsfunktion eines Neurons zu benutzen ist typisch im Bereich des DL [2, vgl. S.31].

Wie kommen nun die Kantengewichte zustande? Den Kanten werden zunächst zufällige Werte als Gewicht zugeordnet. Danach wird jede Eingabe an das Modell übergeben, die daraufhin die einzelnen Layer durchläuft. Die Vorhersage wird nun mit dem tatsächlichen Wert an eine Verlustfunktion übergeben, welche dann den Verlustscore berechnet. Für eine Regression, also die Vorhersage eines numerischen Wertes, bietet sich beispielsweise die Berechnung des mittleren quadratischen Fehlers (mean squared error) an. Dieser Verlustscore wird als Feedback-Signal genutzt und gibt an, wie gut das Modell die Aufgabe lösen kann. Das DL-Modell versucht während des Trainings diesen Wert zu minimieren, indem der Verlustscore an einen Optimierer übergeben wird, der anhand des Verlustscores die Gewichtungen der Kanten im NN aktualisiert. Am Anfang des Trainingsprozesses wird der Verlustscore relativ hoch sein, da die Kantengewichte zufällig ausgewählt wurden. Mit jeder neuen Eingabe werden die Kantengewichte jedoch aktualisiert, was zur Folge hat, dass das Modell immer genauer wird und der Verlustscore somit abnimmt [1, vgl. S.30ff.]. Abbildung 2.8 zeigt dabei grafisch den Aufbau des Trainingsprozesses eines NN. Je mehr Daten vorhanden sind, desto genauer kann das Modell somit werden. Die ImageNet-Datenbank mit ihren 1,4 Millionen Bilddateien hatte beispielsweise einen großen Einfluss auf den Erfolg von DL [1, vgl. S.45]. Wichtig zu beachten ist, dass der Datensatz in einen Trainings- und einen Testdatensatz aufgeteilt werden muss. Ein Modell darf niemals mit den Daten trainiert werden, die auch zum Testen genutzt werden. Wird ein Datensatz nicht aufgeteilt, wird das Modell logischerweise beim Testen sehr gute Ergebnisse erzielen, da es die Testdaten bereits kennt. Der Testdatensatz wird zudem genutzt, um die Genauigkeit des Modells bei unbekannten Daten zu bestimmen. Dieser Schritt ist wichtig, da so geprüft werden kann, wie gut ein Modell mit neuen Daten umgehen kann und ob es in der Lage ist diese korrekt vorherzusagen.

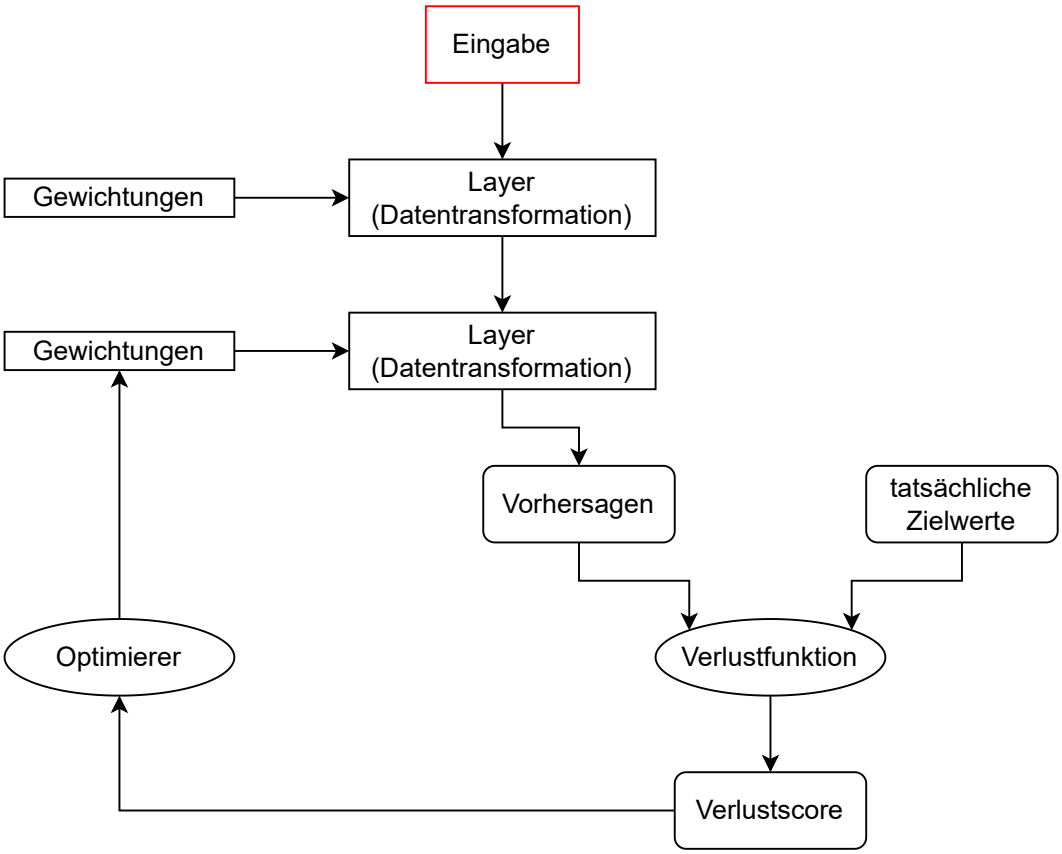


Abbildung 2.8: Anlernen eines neuronalen Netzes [1, S.31]

2.3.3 Overfitting und Underfitting

Overfitting und Underfitting, auf Deutsch Über- und Unteranpassung, können im Trainingsprozesses von ML- sowie DL-Aufgaben auftreten. Unter dem Begriff Underfitting wird verstanden, dass ein Modell noch nicht alle Zusammenhänge im Trainingsdatensatz erkannt und modelliert hat und somit der Verlustscore noch hoch ist, da das Modell für den Trainingsdatensatz noch keine sinnvollen Repräsentationen gefunden hat. Dieses Problem kann gelöst werden, indem die Anzahl an Trainingsdurchläufen erhöht oder die Komplexität des Modells erweitert wird, also weitere Schichten oder mehr Neuronen in einer Schicht hinzugefügt werden. Overfitting beschreibt dabei den Fall, dass ein Model die Eigenheiten der Trainingsdaten sozusagen auswendig lernt und mögliche irrelevante Muster im Trainingsdatensatz erlernt. Das Problem daran ist, dass das Modell dann nicht mehr in der Lage ist, das eigentliche Problem generalisiert zu betrachten und neue Daten korrekt vor-

herzusagen. Mögliche Lösungen wäre dabei mehr Daten zu beschaffen oder die Komplexität des Modells zu verringern. Zudem existiert ein Verfahren, das ebenfalls dabei hilft, Overfitting zu verhindern. Dieses Verfahren nennt sich Dropout-Regularisierung. Bei der Dropout-Regularisierung werden während des Trainings einige zufällig ausgewählte Ausgaben von Neuronen in einer Schicht auf 0 gesetzt. Das soll verhindern, dass ein NN irrelevante Muster im Trainingsdatensatz erlernt [1, vgl. S.142ff.].

Je weniger Daten vorhanden sind und je komplexer ein Modell ist, desto wahrscheinlicher läuft das Modell Gefahr, von Overfitting betroffen zu sein. Je simpler ein Modell ist und je weniger Trainingsdurchläufe durchlaufen werden, desto eher kann es zu einer Unteranpassung des Modells kommen. Es ist also wichtig, ein gutes Mittelmaß zu finden, damit ein Modell in der Lage ist ein Problem verallgemeinert und optimal lösen zu können.

2.3.4 Vor- und Nachteile

Der große Vorteil von DL und NN ist das „Universal Approximation Theorem“, welches besagt, dass jeder funktionale Zusammenhang zwischen Ein- und Ausgabe durch ein NN angenähert werden kann. Das gilt dabei nicht nur für mathematische Zusammenhänge. Voraussetzung dafür ist ein NN, dessen Komplexität groß genug gewählt wird. Dafür kann bereits eine versteckte Schicht ausreichen. Durch diese Eigenschaft kann ein NN theoretisch jedes Approximationsproblem lösen und interessiert sich dabei nicht dafür, ob eine Kausalität zwischen Ein- und Ausgabe besteht und kann ebenfalls mit unvollständigen und falschen Daten arbeiten [2, vgl. S.74ff.]. Zudem sind NN nicht anfällig gegenüber leichten Änderungen im Datensatz, stattdessen sind sie fehlertolerant. Selbst wenn ein Teil des Netzes funktionsunfähig werden sollte, spielt das keine allzu große Rolle, da die Muster und Strukturen, welches ein Modell gelernt hat, auf das gesamte Netz verteilt ist. Diese Tatsache hat zudem den Vorteil, dass wenn neue Daten dazu kommen, nicht das gesamte Netz neu trainiert werden muss, sondern das bestehende Netz upgedatet werden kann [2, vgl. S.82]. Weitere Vorteile von NN sind ihre Einfachheit und Skalierbarkeit. Das Erstellen eines Modells ist einfach und auf eine beliebige Größe skalierbar, wes-

halb NN auch mit großen Datenmengen und einer Vielzahl an Attributen angelernt werden können [1, vgl. S.47].

Neben den Vorteilen bringt die Nutzung von NN auch einige Nachteile mit sich. Zum einen ist die Genauigkeit der Prognose eines Modells stark abhängig von der Qualität der Trainingsdaten. Werden Attribute im Trainingsdatensatz genutzt, die irrelevant bei der Beschreibung des Problems sind, wirkt sich das negativ auf die Genauigkeit des Modells aus, weshalb eine vorhergehende Analyse der Attribute, die sogenannte „Feature Selection“, unausweichlich ist. Zudem kann das Training eines NN mit steigender Komplexität zeit- und rechenintensiv werden, da die Anzahl an zu berechnenden Parametern bei steigender Komplexität zunimmt. Ein komplexeres Modell hat außerdem zur Folge, dass es nicht mehr möglich ist transparent und nachvollziehbar zu Verstehen, wie ein NN eine Ausgabe berechnet. Außerdem muss beim Testen und Validieren eines Modells überprüft werden, ob das Modell nicht eventuell over- oder underfitted und somit nicht in der Lage ist, das zu lösende Problem hinreichend verallgemeinert abzubilden und somit auf neuen Daten zu fehlerhaftem Verhalten neigt [2, vgl. S.84ff.].

Literaturverzeichnis

[1] François Chollet, *Deep Learning mit Python und Keras*. mitp, 2018, vol. 1.

[2] Daniel Sonnet, *Neuronale Netze kompakt : Vom Perceptron zum Deep Learning*. Springer, 2022, vol. 1.

[3] IREB(International Requirements Engineering Board), "Wörterbuch der Requirements Engineering Terminologie," 2022.

[4] Siemens Mobility GmbH, *RM Process Manual - Project Execution*, 2021.

[5] IEEE(The Institute of Electrical and Electronics Engineers), *IEEE Standard Glossary of Software Engineering Terminology*, 1990.

[6] Elizabeth Hull, Ken Jackson, Jeremy Dick, *Requirements Engineering*. Springer, 2005, vol. 2.

[7] ISO/IEC/IEEE International Standard, "29148-2018 - Systems and software engineering – Life cycle processes – Requirements engineering," 2018.

[8] The Standish Group, "The CHAOS Report," 1994.

[9] Siemens Mobility GmbH, *RM Process Manual - Application Rules*, 2022.

[10] IBM, *The DXL Reference Manual*, https://www.ibm.com/docs/en/SSYQBZ_9.5.0/com.ibm.doors.requirements.doc/topics/dxl_reference_manual.pdf, 2012, accessed: 2023-03-01.

[11] OpenAI, "Introducing ChatGPT," https://openai.com/blog/chatgpt, accessed: 2023-03-02.

[12] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. 59, no. 236, pp. 433–460, 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433

[13] Huawei Technologies Co., Ltd, *Artificial Intelligence Technology*. Springer, 2023.