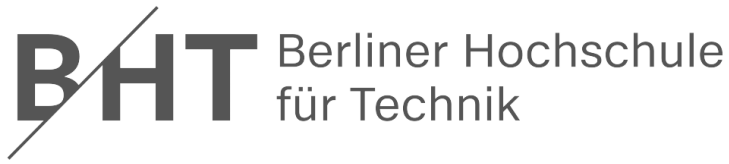
Evaluierung und Optimierung von Large Language Models für die Entwicklung von Webanwendungen

Ein Ansatz zur Verbesserung des Entwicklungsprozesses bei Softwareprojekten



Masterthesis
für den Master of Science Studiengang Medieninformatik

eingereicht von: Wilfried Pahl
Matrikelnummer: 901932
Studiengang: Online Medieninformatik
Berliner Hochschule für Technik

betreut durch Prof. Dr. S. Edlich
Berliner Hochschule für Technik

Temmen-Ringenwalde, der 28. Oktober 2024

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel „Evaluierung und Optimierung von Large Language Models für die Entwicklung von Webanwendungen (*Ein Ansatz zur Verbesserung des Entwicklungsprozesses bei Softwareprojekten*)“ selbstständig und ohne unerlaubte Hilfe verfasst habe. Alle benutzten Quellen und Hilfsmittel sind vollständig angegeben und wurden entsprechend den wissenschaftlichen Standards zitiert.

Ich versichere, dass alle Passagen, die nicht von mir stammen, als Zitate gekennzeichnet wurden und dass alle Informationen, die ich aus fremden Quellen übernommen habe, eindeutig als solche kenntlich gemacht wurden. Insbesondere wurden alle Texte und Textpassagen anderer Autoren sowie die Ergebnisse von Sprachmodellen wie OpenAI's GPT-3 entsprechend den wissenschaftlichen Standards zitiert und referenziert.

Ich versichere weiterhin, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe und dass ich keine Teile dieser Arbeit in anderer Form für Prüfungszwecke vorgelegt habe.

Mir ist bewusst, dass eine falsche eidesstattliche Erklärung strafrechtliche Konsequenzen haben kann.

Temmen-Ringenwalde, den 28. Oktober 2024

Unterschrift

Abstract

Abstract in Englisch.

Zusammenfassung

Zusammenfassung in Deutsch.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

CoT Chain of Thoughts.

DL Deep Learning.

k-NN k-Nearest Neighbors.

KI Künstliche Intelligenz.

KNN Künstliche Neuronale Netze.

LLM Large Language Model.

MAS Multi-Agenten-System.

ML Maschine Learning.

NLP Natural Language Processing.

RBF Radial Basis Function.

ToT Tree of Thoughts.

Einleitung

1

1.1 Hintergrund und Kontext

Durch die zunehmende Globalisierung und Digitalisierung wird die Gesellschaft der Gegenwart und Zukunft geprägt. Der Ausbau von Hochgeschwindigkeitsnetze und die globale Corona-Pandemie haben diese Entwicklung noch einmal beschleunigt. Immer mehr Unternehmen erkennen die Potenziale der Digitalisierung und stellen ihre Geschäftsprozesse um. Ganze Wertschöpfungsketten werden auf cloudbasierte Umgebungen umgestellt. Angefangen bei der Kommunikation, über Beschaffung und Produktion bis zum Verkauf der Waren und Dienstleistungen, vergleiche mit [1, Seite 21 ff.] und [2]. In allen Stufen der Prozesse kommen webbasierte Anwendungen zum Einsatz, um die Kommunikation der Anwender mit den Systemen zu ermöglichen oder Schnittstellen für die Datenübertragung zwischen den verschiedenen Systemen zu gewährleisten. Durch wachsende Anzahl von Web-Anwendungen wächst auch der Druck für die Entwicklungsfirmen, ihre Anwendungen den schnell und oft wechselnden Kundenanforderungen anzupassen.

Durch diesen Prozess getrieben, müssen Entwicklungsfirmen in immer kürzeren Release-Zyklen Softwarekomponenten hinzufügen und vorhandene erweitern. Gleichzeitig wachsen aber auch die Anforderungen an Stabilität und Sicherheit der cloudbasierten Anwendungen, sowie der Bedarf an kostengünstigeren IT-Abläufen (Beweis fehlt). Ein weiteres Problem ist der wachsende Fachkräftemangel in der Wirtschaft und die damit verbundenen steigenden Gehälter der Entwickler (Beweis fehlt).

Die Verwendung künstlicher Intelligenz bei der Programmierung gewinnt immer mehr an Bedeutung. Eine Technologie die im besonderen Maße an dieser Entwicklung beteiligt ist, sind die Large Language Models. Insbesondere mit der Veröffentlichung vom ChatGPT wurde hier ein regelrechter Hype um die LLMs ausgelöst. Diese Modelle erlauben eine Softwareentwicklung mit natürlicher

Sprache. Tiefe Kenntnisse der verwendeten Programmiersprache sind nicht mehr in dem Maße erforderlich, wie ohne LLMs.

1.2 Problemstellung

So groß der Hype um Künstliche Intelligenz auch sein mag, zurzeit kann KI nicht alle Anforderungen selbstständig lösen. Dies sollte auch bei der Verwendung von KI generierten Inhalten und Programmcodes beachtet werden.

> *KI denkt nicht, KI trifft keine Entscheidungen. Eine KI antwortet auf eine Eingabe nicht mit der besten Antwort, sondern mit der Wahrscheinlichsten.*
>
> Vattenfall Online , KI für Unternehmen – die Grenzen der KI

Der Mensch muss die generierten Ergebnisse überprüfen, ehe erstellte Programmcodestücke in vorhandene Programme eingefügt und in Produktionsumgebungen implementiert werden.

Viele Entwickler setzen auf Chatbots, wie ChatGPT oder Gemini zur Generierung von Code, wie eine Umfrage von *stackoverflow* vom Mai 2024 zeigt [3]. Gleichzeitig wachsen auch die technischen Schulden bei Softwareprojekten, da diese Modelle nicht für die Entwicklung von Software optimiert sind (Beweis fehlt).

1.3 Zielsetzung und Forschungsfragen

Das Ziel in der Softwareentwicklung war und ist die Optimierung des Entwicklungsprozesses, um Ressourcen und Kosten einzusparen und dadurch einen Wettbewerbsvorteil zu erlangen. Die steigende Nachfrage von Cloud-Anwendungen, steigt auch der Optimierungsdruck in diesem Bereich besonders stark.

Vor diesem Hintergrund lässt sich die Zielsetzung bereits aus dem Titel „*Evaluierung und Optimierung von Large Language Models für die Entwicklung von Webanwendungen*“ dieser Arbeit herleiten. Diese Arbeit soll eine Auswahl von Modellen evaluieren und dessen Brauchbarkeit für die Softwareentwicklung aufzeigen. Um die Antworten der Modelle zu optimieren, soll eine Evaluation von Methodiken erfolgen, bei der deren Anwendung auf die Modelle eine Verbesserung der Antworten ersichtlich ist. Des Weiteren soll gezeigt werden, ob und wie weit sich der Prozess der Codegenerierung automatisieren lässt und ob einige

Programmiersprachen, die in der Webentwicklung Verwendung finden, besser unterstützt und geeignet sind als andere und somit zu bevorzugen sind.

Die vier Ziele dieser Arbeit lassen sich in den folgenden kurz formulierten Sätzen zusammenfassen,

Z1 Welche Modelle eigenen sich für die Softwareentwicklung.

Z2 Welche Methodiken helfen die Qualität der Antworten von Modellen zu verbessern.

Z3 Wie weit lässt sich die Verwendung von großen Sprachmodellen, für die Erstellung von Webanwendungen automatisieren.

Z4 Sind einige Programmiersprachen für die Codegenerierung besser geeignet als andere.

1.4 Aufbau der Arbeit

Ein paar Worte zum Aufbau dieser Arbeit. Um ein grundlegendes Verständnis für diese Arbeit zubekommen, werden im Kapitel 2 die Grundlagen besprochen.

Im Kapitel 3 wird der aktuelle Stand der Forschung vorgestellt und Erkenntnisse anderer Arbeiten diskutiert. Die in dieser Arbeit verwendetet Methoden, werden im Kapitel 4 behandelt. Die Implementierung der Test LLMs wird in Kapitel 5 besprochen und in Kapitel 6 die Ergebnisse evaluiert. Bevor in Kapitel 9 auf mögliche Folgearbeiten eingegangen wird, gibt es in Kapitel 7 Anwendungsszenarien, die zu den Ergebnissen dieser Arbeit geführt haben.

1.5 Abgrenzung

In dieser Arbeit fokussiert sich die Betrachtung auf den Bereich der Webanwendungsentwicklung und deren verwendete Programmiersprachen. Parallelen zu anderen Anwendungsbereichen, wie beispielsweise Desktop-Anwendungsentwicklung werden hier nicht expliziert betrachtet können aber durchaus vorkommen.

Auch wenn rechtliche und ethische Überlegungen einen wichtigen Aspekt in Umgang mit Künstlicher Intelligenz darstellt, wird dies in dieser Arbeit nicht betrachtet. Es gibt hinreichend Literatur zu diesen Themen, die in dieser Arbeit Beachtung finden, es wird aber nicht explizit darauf eingegangen.

Grundlagen

2

Die hier besprochenen Grundlagen gehen nicht in eine Tiefe, um alle evtl. Fragen zu klären. Jedes einzelne Gebiet könnte eine Arbeit füllen. Stattdessen soll hier lediglich ein kleiner Einblick geben werden.

2.1 Künstliche Intelligenz

Die Grafik 2.1 soll die Einordnung der besprochenen Begriffe im Bereich der Künstlichen Intelligenz (KI) zeigen.
In den folgenden Kapiteln werden die wichtigsten Begriffe und Technologien erläutert.

Eine explizite Definition für *künstliche Intelligenz* ist zurzeit noch nicht einheitlich erfolgt. Geschuldet ist diese Tatsache, dass der Begriff *Intelligenz* nicht eindeutig definiert ist. Somit finden sich viele Versuche eine Definition für künstliche Intelligenz herzuleiten. Ein Ansatz Intelligenz zu erklären und definieren kommt von Jean Piaget[1], welche in dieser Arbeit verwendet wird,

> *Intelligenz ist ein Zustand der Balance oder des Gleichgewichts, der durch eine Person erreicht wird, wenn sie dazu fähig ist, angemessen mit den ihr vorliegenden Daten umzugehen. Aber sie [d.h. Intelligenz] ist kein statischer Zustand, sondern in dem Sinne dynamisch, dass sie sich selbst kontinuierlich an neue Umweltreize anpasst*
>
> Jean Piaget

In dieser Arbeit wird als Definition für die Künstliche Intelligenz, die Definition aus [4, 6 ff.] verwendet.

[1] Jean Piaget war ein Schweizer Biologe und Pionier der kognitiven Entwicklungspsychologie. Er lebte von 9.Aug. 1896 bis 16. Sep. 1980

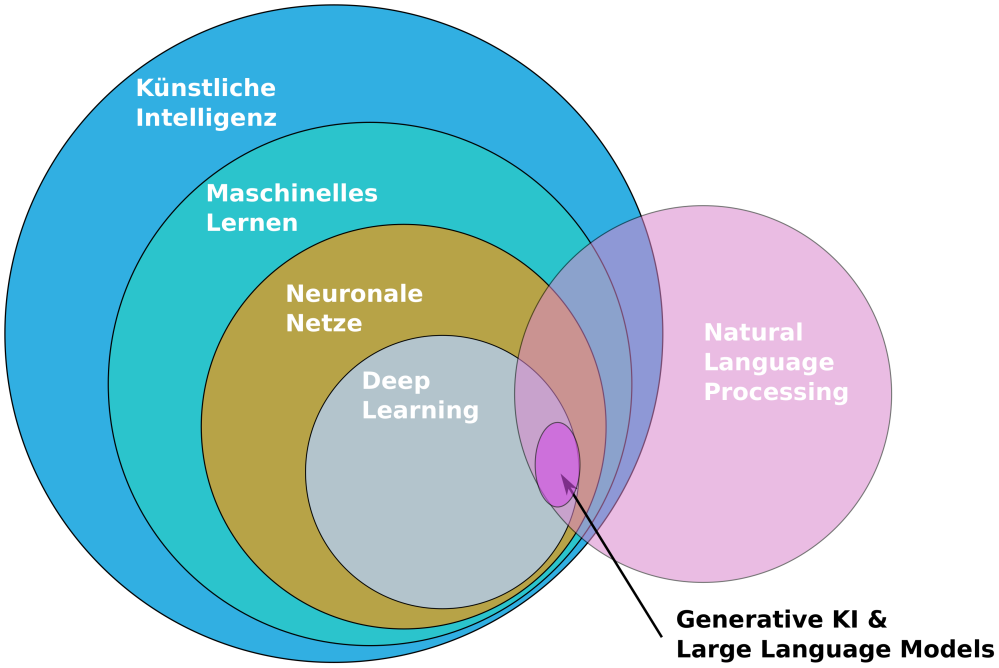


Abbildung 2.1: Large Language Models im Kontext von Künstlicher Intelligenz

Systeme der künstlichen Intelligenz (KI-Systeme) sind vom Menschen entwickelte Softwaresysteme (und gegebenenfalls auch Hardwaresysteme), die in Bezug auf ein komplexes Ziel auf physischer oder digitaler Ebene handeln, indem sie ihre Umgebung durch Datenerfassung wahrnehmen, die gesammelten strukturierten oder unstrukturierten Daten interpretieren, Schlussfolgerungen daraus ziehen oder die aus diesen Daten abgeleiteten Informationen verarbeiten, und über das bestmögliche Handeln zur Erreichung des vorgegebenen Ziels entscheiden. KI-Systeme können entweder symbolische Regeln verwenden oder ein numerisches Modell erlernen, und sind auch in der Lage, die Auswirkungen ihrer früheren Handlungen auf die Umgebung zu analysieren und ihr Verhalten entsprechend anzupassen.

2.1.1 Historisches

An dieser Stelle eine kleine historische Exkursion in der Entwicklung der künstlichen Intelligenz.

1966: Rückschläge bei der maschinellen Übersetzung brachten die Entwicklung nahezu zum Erliegen, lediglich einige wenige Gruppen. Durch die Bedrohungsszenarien des *Kalten Krieges* wurden von der

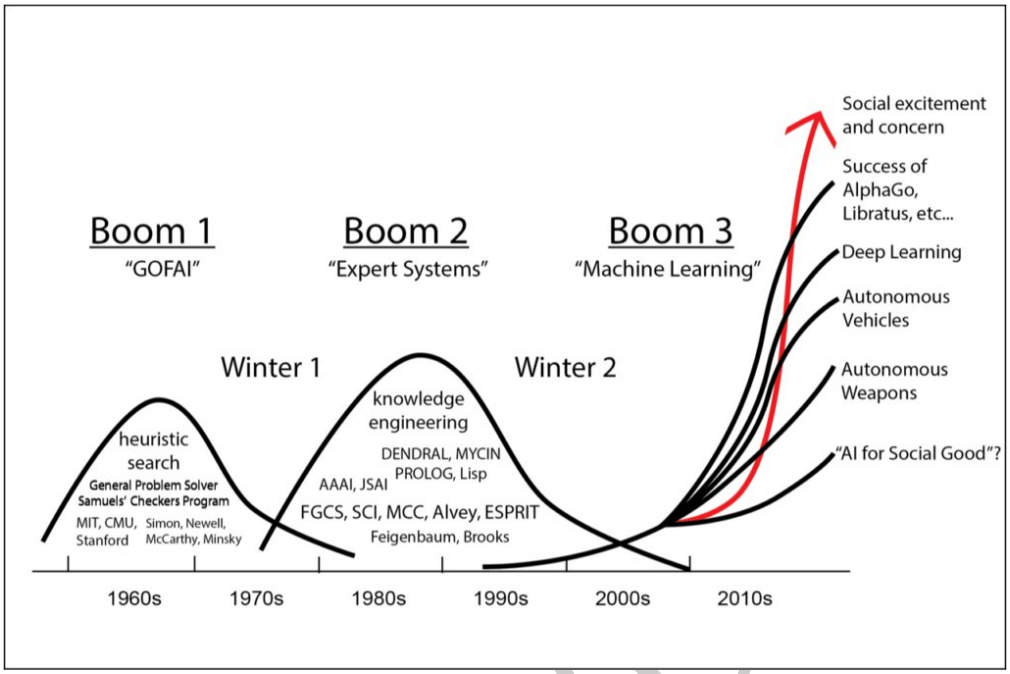


Abbildung 2.2: KI Winterzyklen

US-Regierung und Militär Unsummen von Geldern bereitgestellt, um die Forschung von maschinellen Übersetzungen voran zutreiben. Durch die ausbleibenden Erfolge wurden diese Gelder jedoch gestrichen.

Frank Rosenblatt führte 1958 das Perzeptron ein, das in der Lage war einfache logische Operationen in Neuronalen Netzen auszuführen. Ein Problem was diese Netzwerke nicht lösen konnten war die *nicht lineare Trennbarkeit*, wie beispielsweise das XOR-Problem. Dieses Problem wurde in der Arbeit von Marvin Minskys und Seymour Papert gezeigt. Diese Arbeit beeinflusste die Entwicklung so massiv, dass **1969: die Auswirkungen der Perzeptron-Kritik** dazu betrugen den ersten KI-Winter auszulösen.

1971-75: Die Finanzierungskürzung der DARPA (Defense Advanced Research Projects Agency), die aufgrund der Unzufriedenheit, bei den Fortschritten natürliche Sprache in realen Szenarien, effektiv durch Systeme verarbeiten zu können, trug ebenfalls zur Auslösung des ersten KI-Winters bei. Diese Entscheidung übertrug sich auch auf andere Bereiche in der KI Entwicklung.

1973: Der Fallout des Lighthill-Berichts war der entscheidende Faktor, das den KI-Winter auslöste. Dieser Bericht bewertet die wies auf erhebliche Mängel in den Bereichen Robotik und Sprachverarbeitung hin. Der Bericht führte zu einer erheblichen Skepsis gegenüber der KI und in vielen Projekten zur Kürzung der Fördermittel. Er wirkte sich auch auf die Moral und das Vertrauen der Community aus, welche aus diesem Grund nur sehr kleine und vorsichtige Fortschritte in der KI Forschung erzielte.

Der **1987: Zusammenbruch der LISP-Maschine** markierte den Beginn des *zweiten KI-Winters*. Die spezialisierten LISP-Maschinen, erfuhren einen Nachfragerückgang, ausgelöst durch neu entwickelten Allzweckcomputer von IBM und Apple. Dieses Ereignis zeigt auch beispielhaft wie technologische und wirtschaftliche Faktoren zur Neubewertung und Neuausrichtung ganzer Fachrichtungen führen können.

Die **1988: Strategischen Kürzungen im Computerbereich** durch die US-Regierung trugen ebenfalls zum Ausbruch des *zweiten KI-Winters* bei. Auslöser waren die enttäuschenden Ergebnisse und unerfüllten Erwartungen der Forschung in verschiedenen KI Projekten. Dadurch erfolgte eine Neubewertung der Investitionspriorisierung und führte zu einer deutlichen Kürzung der Gelder. Diese Bewertung wirkte sich auch auf viele weitere Projekte der KI-Entwicklung aus, sodass diese ganz aufgegeben und neue Technologien erforscht und entwickelt wurden.

In den **1990er Jahre: war der Niedergang von Expertensystemen** ebenfalls ein Auslöser, der dazu betrug, dass die Entwicklung von KI-Systemen stagnierte. Deren Wartung war sehr kosten- und arbeitsintensiv, ebenfalls wurden die ständig aktuell benötigten Datenbanken nicht automatisiert aktualisiert. Auch erfüllten sie nicht die in ihnen gesetzten Erwartungen.

In den Jahren nach 2000 bis 2020 trat ein bedeutender Wandel bei den KI-Technologien ein, der als **KI-Frühling des frühen 21. Jahrhunderts** gesehen wird. Besonders die Fortschritte im Bereich *maschinelles Lernen*, *Deep Learning*, und *neuronale Netze* sorgten für neue Investitionen in diesen Bereichen. Besonders namhafte Tech-Firmen, wie IBM, Google und Microsoft stellten große Budges für die Forschung. Gefördert wurde die Entwicklung durch neue leistungsfähigere Rechner und Daten die in großen Mengen zur Verfügung standen. Hinzu kam, dass KI in die Mainstream-Technologie integriert und in viele Geschäftsprozesse Einzug hielt.

2.1.2 Maschinelles Lernen

Das Gebiet um das Maschinelle Lernen (ML) ist ein Teilgebiet der Künstlichen Intelligenz. Für das maschinelle Lernen wird in dieser Arbeit die allgemein gültige Definition nach Tom M. Mitchell verwendet.

> *A computer programm is said to learn from experience E with respekt to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*
>
> Ein Computerprogramm lernt aus Erfahrung E in Bezug auf eine Klasse von Aufgaben T und ein Leistungsmaß P, wenn sich seine Leistung bei Aufgaben in T, gemessen an P, mit der Erfahrung E verbessert.
>
> Mitchell, Tom M
> Machine Learning

Bei dieser Definition ist,
E die **Erfahrung**, die Daten aus denen das System lernt,
T beschreibt die **Aufgabe**, die das System erledigen soll und
P ist die **Leistungskennzahl** an dem der Erfolg des Systems die Aufgabe zu lösen gemessen wird.

ML und KI sind nicht wirklich in der Lage selbstständig zu lernen oder denken, sie imitieren dies lediglich. ML ist aber wohl in der Lage komplexe Muster und Funktionen in großen Datenmengen zu erkennen. Durch die Unfähigkeit zu lernen sind KI Technologie nicht in der Lage neuen Inhalte zu schaffen.

Es ist auch egal wie gut die Modelle trainiert werden, eine 100% Fehlerfreiheit gibt es nicht. Für die Praxis bedeutet dies, die Ausgaben von Modellen müssen durch Menschen immer wieder evaluiert werden, um dessen Richtigkeit sicherzustellen. Zudem sollten keine Modelle verwendet werden, wenn die Lösung eines Problems durch einen konkreten Algorithmus erfolgen kann. Hier können die Ergebnisse nachvollzogen werden und sind für den Menschen besser zu verstehen.

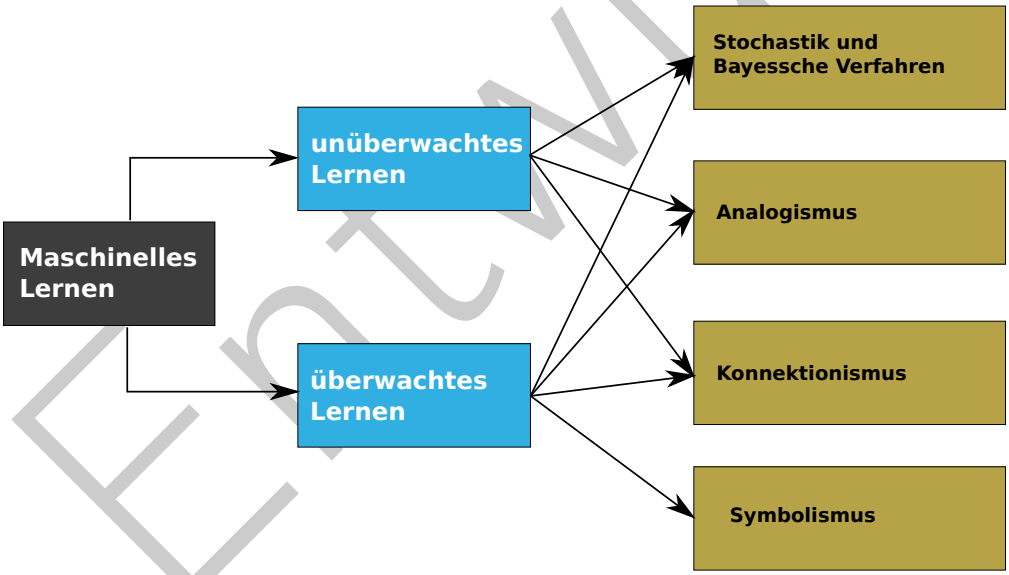


Abbildung 2.3: Lernparadigmen des maschinellen Lernens

2.1.3 Lernparadigmen des ML

Das maschinelle Lernen wird in zwei wichtige Formen der Lernparadigmen unterteilt. Dem überwachten (supervised Learning) und unüberwachten Lernen (unsupervised Learning). Abbildung 2.3 zeigt die wichtigsten Lernparadigmen des maschinellen Lernens und dessen Methodiken.

Überwachtes Lernen

Bei überwachten Lernen sind für die Eingaben der Trainingsdaten dazugehörige Ausgaben (Labels) definiert. Das Ziel ist es eine Funktion zu trainieren um künftige Eingaben korrekt klassifizieren oder vorhersagen zu können.

Unüberwachtes Lernen

Bei unüberwachten Lernen sind die gelabelten Ausgaben nicht vorhanden. Hierbei wird beispielsweise durch Clustering oder Dimensionsreduktion versucht Muster und Strukturen zu erkennen.

2.1.4 Theoretische Grundlagen des Lernens

In den folgenden Kapiteln werden vier wichtige Konzepte zum maschinellen Lernen besprochen.

Stochastik und Bayessches Theorem

Als Teilgebiet der Mathematik befasst sich die Stochastik mit Wahrscheinlichkeitsverteilung und zufälligen Prozessen. Auf dem Gebiet des maschinellen Lernens werden mithilfe der Stochastik Prognosen erstellt. Es wird versucht auf der Basis vorhandener Daten, bei neuen Daten eine Wahrscheinlichkeitsverteilung vorherzusagen. Durch geeignete Modellierung wird versucht einen kontrollierten Umgang mit Unsicherheiten zu erlangen.

Bei den Bayesschen Theorem handelt es sich um stochastische Methoden, die auf dem Bayessches Theorem basieren. Mit dessen Hilfe soll auf Basis der vorliegenden Daten das beste Modell, für die Vorhersage gefunden werden. Das Verfahren berücksichtigt immer die neusten Daten, um die Wahrscheinlichkeitsverteilung zu aktualisieren. Siehe Formel 2.1

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.1}$$

Analogismus

Dieser Lernansatz sucht nach Ähnlichkeiten in den Daten. Er basiert auf der Annahme, dass ähnliche Daten ähnliche Vorhersagen oder Klassifizierungen besitzen. Dieses Modell lernt, indem neue Daten mit bekannten vergleicht und nach ähnlichen Strukturen und Mustern sucht. Ein bekanntes Verfahren für diesen Lernansatz ist der k-Nearest Neighbors (k-NN).

Der Analogismus wird im überwachten Lernen als auch im unüberwachten Lernen angewandt, um Muster und Strukturen zuerkennen.

Konnektionismus

Der Lernansatz des Konnektionismus beruht auf kleine Einheiten die miteinander verbunden sind. Diese werden als Neuronen bezeichnet, die den Nervenzellen von Organismen nachempfunden sind. Die Künstlichen neuronalen Netze sind die bekanntesten Vertreter auf denen auch Deep Learning Modelle basieren.

Auch dieser Lernansatz wird im unüberwachten und überwachten Lernen angewandt.

Symbolismus

Anders als beim Konnektionismus arbeiten die Einheiten beim Symbolismus mit explizite formale Regeln und Symbole, um das Wissen darzustellen. Der Symbolismus ist weniger flexible im Umgang mit unvollständigen Datensätzen und Unsicherheiten. Daher hat dieser Lernansatz weniger Relevanz als der Konnektionismus.

Dieser Lernansatz findet im überwachten Lernen Anwendung, als Beispiel sind hier Entscheidungsbäume zu nennen.

2.1.5 Neuronale Netze

Neuronale Netze oder auch künstliche neuronale Netze (KNN) sind spezifische Typen des maschinellen Lernens. Sie sollen die biologischen Neuronen des Gehirns nachempfinden. Die Abbildung 2.4 von [5] zeigt eine stark vereinfachte biologische Nervenzelle.
Bei Nervenzellen werden elektrische Eingangssignale über Dendriten aufgenommen und in den Zellkern geleitet. Dort werden die eingehenden Signale zusammen geführt und es bildet sich das Aktionspotential. Übersteigt das Aktionspotential das Schwellenpotential der Zelle, so wird das Signal über das Axon abgeleitet, die Nervenzelle „*feuert*“.

Bevor das Signal die nächste Nervenzelle reizen kann, muss es über den synaptischen Spalt. Hierbei ist die Präsynapse das Axon der sendenden Zelle, die Postsynapse ist ein Dendrit der empfangenden Nervenzelle. Der Reiz wird durch die Botenstoffe Dopamin oder Serotonin übertragen. Dopamin wirkt als Verstärker der Reizübertragung, während Serotonin hemmend wirkt.

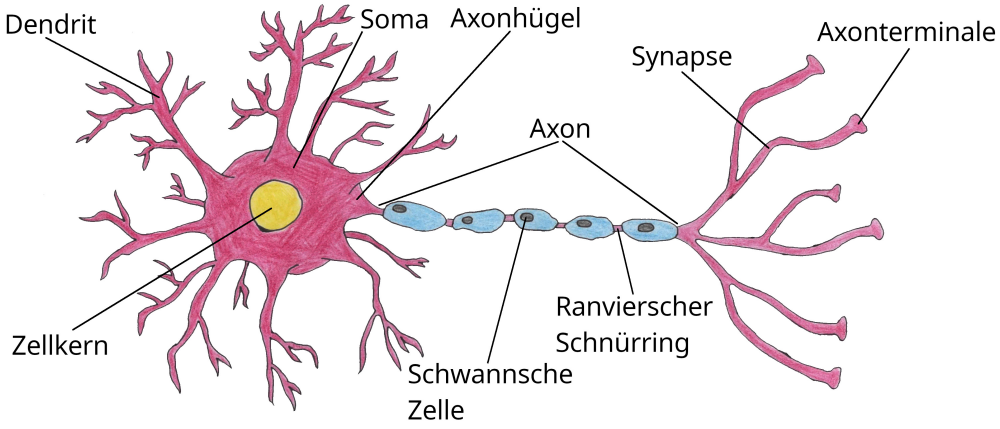


Abbildung 2.4: Biologische Nervenzelle

Neuronen im neuronalen Netz

Die kleinste Einheit in künstlichen neuronalen Netzen sind die Neuronen. Sie sind den biologischen Nervenzellen nachempfunden.

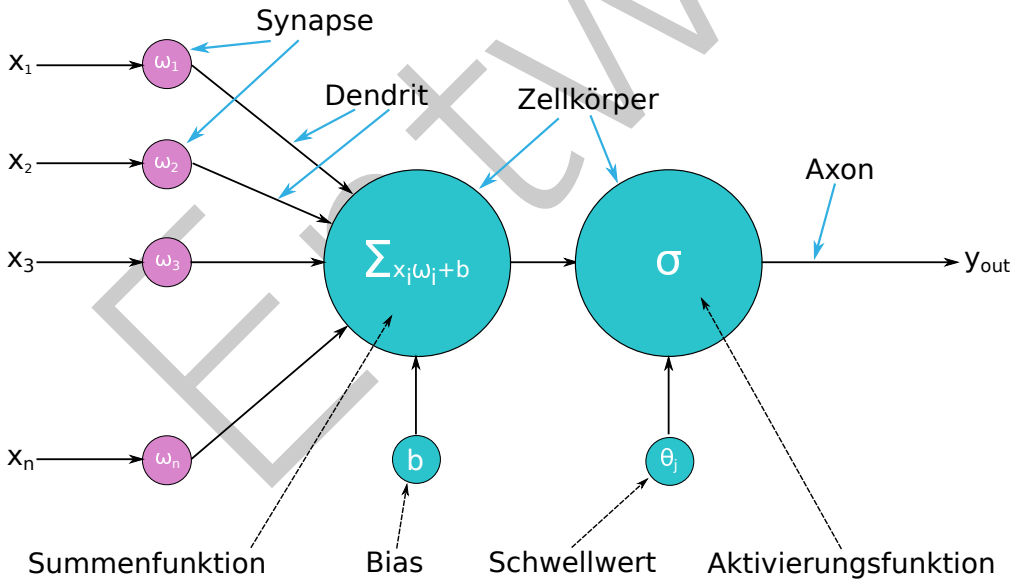


Abbildung 2.5: Künstliche Nervenzelle

Sie haben als Eingangswert einen Vektor und als Ausgangssignal ein Skalar. Außer in der Eingabe Schicht ist jedes Eingangssignal x_n ein Ausgangssignal y_{out} eines anderen Neuron. Die Wichtungen der Eingangssignale modellieren den synaptischen Spalt zwischen zwei biologischen Nervenzellen. Dieser kann

ebenfalls verstärkten oder hemmend wirken. Alle Eingangssignale zusammen mit den Wichtungen, werden durch die Summenfunktion aufaddiert. Im Anschluss wird das Bias mit eingerechnet. Die Formel 2.2 zeigt die Summenfunktion für n Eingangssignale mit Beachtung des Bias Wert.

$$y = x_1 + x_2 + \cdots + x_n + b \tag{2.2}$$

Durch das Bias wird die Berechnung in der Summenfunktion flexibler, eine Verschiebung in Richtung der y-Achse ist möglich, ohne die Wichtungen anzupassen und es ermöglicht eine Ausgabe auch, wenn alle Eingänge $Null$ sind.

Nach der Summenfunktion wird das Signal an die Aktivierungsfunktion übergeben. Diese Funktion leitet ein Signal erst weiter, wenn ein festgelegter Schwellwert überschritten wird. Die Analogie zur biologischen Nervenzelle ist das Aktionspotential, welches durch die Reize anderer Nervenzellen aufgebaut wird und wie beim künstlichen Neuron führt das Überschreiten eines Schwellenwertes dazu, dass das Neuron „feuert“.

Je nach Problem ist es wichtig die richtige Aktivierungsfunktion zu wählen. In [6] wird beschrieben wie dies erfolgen kann, hier ist die Abbildung 2.6 entnommen.

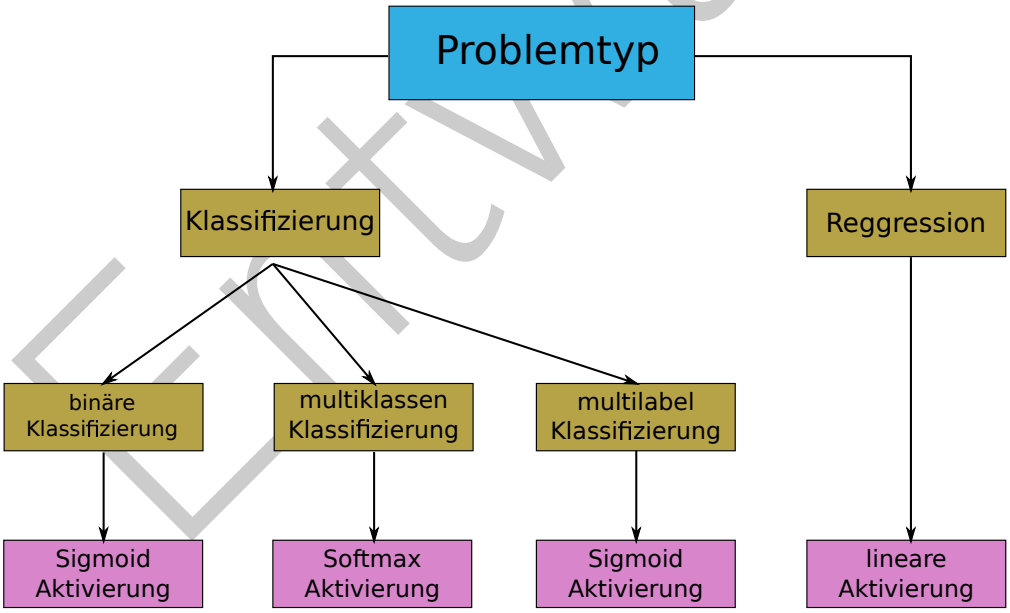


Abbildung 2.6: Auswahl der richtigen Aktivierungsfunktion

Im Folgenden werden einige Aktivierungsfunktionen vorgestellt.

Identity-Aktivierungsfunktion

Die Gleichung der Identity Funktion ist in Formel 2.3 zu sehen. Da die Ausgabe der Eingabe entspricht,

sind alle Werte möglich die in der Summenfunktion entstehen. Aus ihr geht hervor das der Wertebereich theoretisch von $-\infty$ bis ∞ reicht.

Einsatz findet die Identity Funktion beispielsweise in der Ausgabeschicht, wenn die Ausgabe lineare oder direkt proportionale zur Eingabe geben soll. Dies kann bei Regressionsaufgaben der Fall sein. Eine weitere Möglichkeit ist der Einsatz in den Stellen im neuronalen Netz bei der die Ausgabe einfach nur weiter gegeben werden muss, ohne diese zu ändern. Ebenfalls kann die Identity Funktion als Grundlage für kombinierte Aktivierungsfunktionen dienen.

$$\sigma(x) = x \tag{2.3}$$

Der Graph der Identity Funktion und dessen Ableitung ist in Abbildung 2.7 dargestellt.

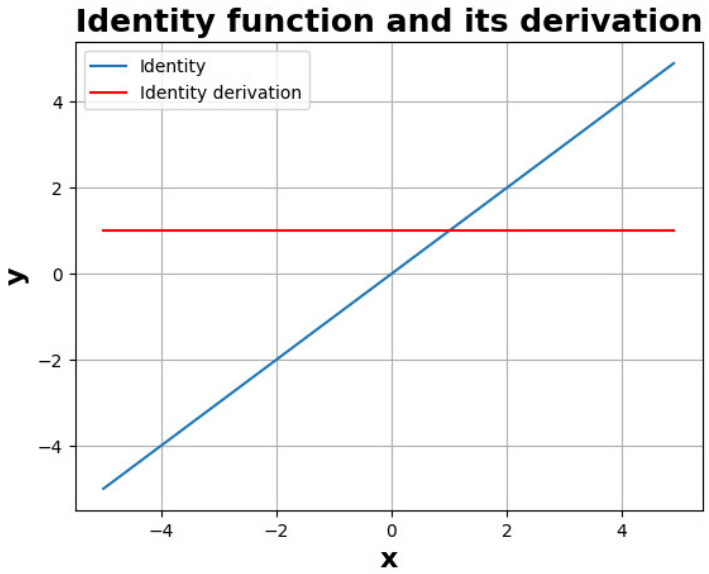


Abbildung 2.7: Graph Identity Funktion und deren Ableitung

Die Identity ist eine sehr einfache Funktion, bei der der Funktionswert gleich dem Eingangswert ist. Die Abbildung 2.7 zeigt die Ableitung der Funktion, die immer den Wert eins annimmt und somit streng monoton steigend ist. Dadurch ist die Anpassung der Gewichte durch Backpropagation sehr einfach.

Binary Step

Wie in [7, S. 311–312] beschrieben, kann die binäre Step-Aktivierungsfunktion eingesetzt werden, wenn es schwellen basierte Klassifizierung geht. Sie eignet sich nicht für Multiklassen-Klassifizierung.
Der Funktionswert der Gleichung kann entweder $eins$ oder $null$ annehmen. Die Formel 2.4 zeigt diese

Definition.

$$\sigma(x) = \begin{cases} 1 & : \ x \geq 0 \\ 0 & : \ x < 0 \end{cases} \tag{2.4}$$

Die Abbildung 2.8 zeigt den Funktionsverlauf und die Ableitung dieser Funktion. Die Funktion liefert entweder $null$ bei negativen Zahlen oder $eins$ bei positiven Zahlen.

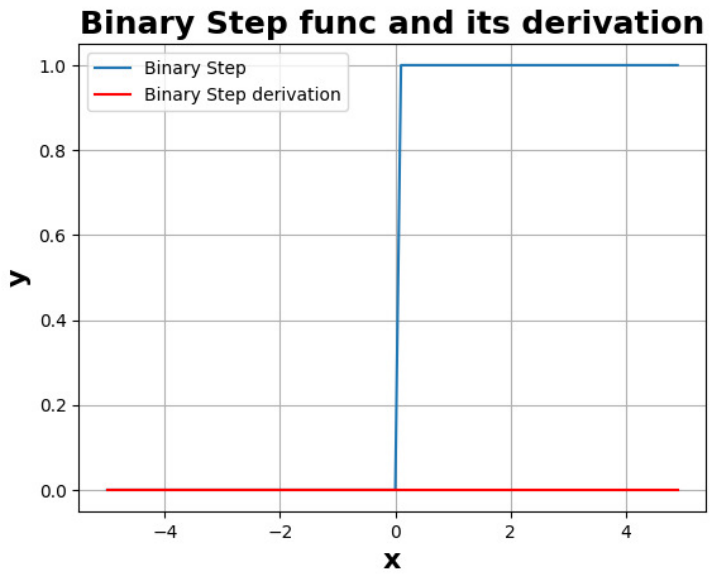


Abbildung 2.8: Graph der Funktion *Binary Step* und deren Ableitung

Ein Problem, welches ebenfalls in [7, S. 311–312] beschrieben wird, ist das der Gradient für x Null ist. Dieser Umstand kann dazu führen, dass die Backpropagation nicht ausgeführt werden kann.

Sigmoidfunktion

Die Sigmoidfunktion ist eine der bekanntesten Aktivierungsfunktionen und eine Sonderform der logarithmischen Funktion. Ein Eingangswert x liefert einen garantierten Ausgabewert zwischen $null$ und $eins$. Sie ist monoton steigend und überall differenzierbar. Der Einsatz der Sigmoidfunktion als Aktivierungsfunktion in einer Hidden-Schicht ermöglicht dem neuronalen Netz eine nicht lineare Trennbarkeit im 2-dimensionalen zu lernen. Ebenfalls ermöglichen ihre Eigenschaften, bei kleinen Netzwerktiefen den Backpropagation-Algorithmus einfach anzuwenden und ermöglicht das Lernen des Netzwerks. Bei großen Netzwerken mit vielen Schichten kann es aber zum „Problem der verschwindenden Gradienten“ führen auch bekannt als *Vanishing-Gradient-Problems*. Es ist bekannt, dass die Ableitung der Sigmoidfunktion einen Maximalwert von $0,25$ annehmen kann. Bei mehreren Schichten wird das Produkt der Ableitung immer kleine bis dieser

sich $null$ annähert.

Die Formel 2.5 zeigt die Sigmoidfunktion, die Formel 2.6 die erste Ableitung der Sigmoidfunktion.

$$\sigma(x) = sig(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

$$\sigma'(x) = sig(x) * (1 - sig(x)) \tag{2.6}$$

Der Graph der Sigmoidfunktion und ihre Ableitung ist in Abbildung 2.9 dargestellt. Hierbei ist zu erkennen, dass die Ableitung eine Symmetrie im Ursprung zeigt. Im Ursprung die maximale Steigung zu finden ist und für kleine und große x-Werte gegen $null$ konvergiert.

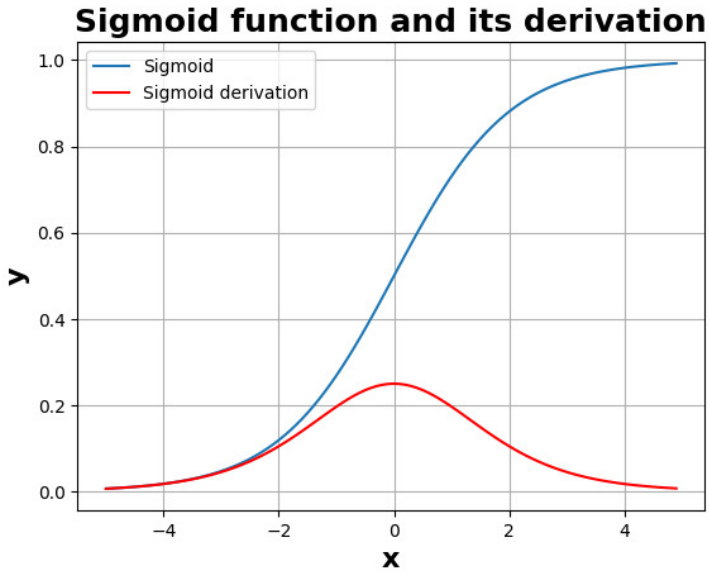


Abbildung 2.9: Graph Sigmoidfunktion und deren Ableitung

Diese Funktion hat zwei Nachteile. Zum einen bei kleinen und großen Werten, konvergiert der Gradient gegen null. Somit wird die Verlustfunktion sehr klein und die Aktualisierung der Gewichte während des Lernens wird verhindert.

Tangens Hyperbolicus

Ebenso wie die Sigmoidfunktion ist auch die Tangens Hyperbolicus eine monoton stetig steigende Funktion, die symmetrisch zum Ursprung ist. Der Ausgabewert liegt garantiert hier zwischen -1 und 1, was dazu beiträgt, dass die Ausgabe der Schicht um eins zentriert bleibt, welches ein Vorteil bei Normalisierung der

Ausgabewerte ist. Auch hier haben wir, in tiefen Netzen das „Problem des verschwindenden Gradienten“ wie bei der Sigmoidfunktion. In flachen Netzen erleichtern die Eigenschaften den Optimierungsprozess.

$$\sigma(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x - e^{-x}} \tag{2.7}$$

$$\sigma'(x) = tanh^2(x) \tag{2.8}$$

Die Abbildung 2.7 zeigt den Graphen der Tangens Hyperbolicus Funktion und deren Ableitung. Deutlich zu erkennen ist die Symmetrie zum Ursprung bei der Ableitung, hier rot dargestellt.

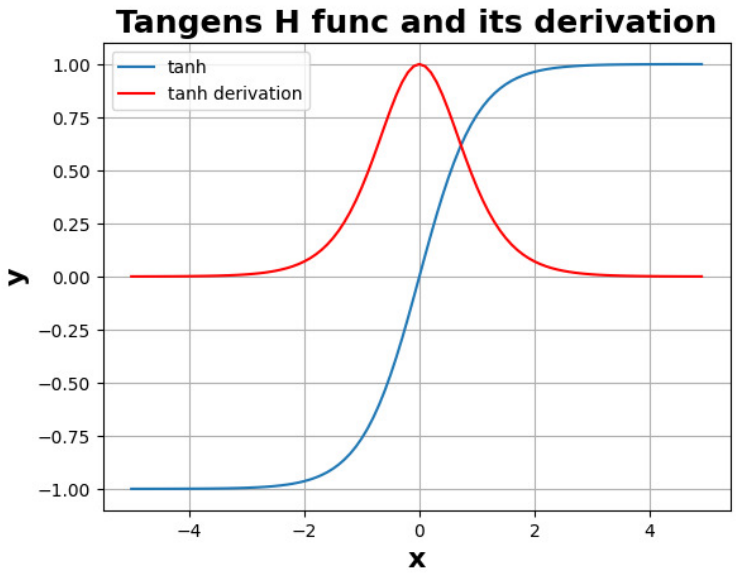


Abbildung 2.10: Graph Tangens Hyperbolicus und deren Ableitung

Die Verwendung der Tangens Hyperbolicus als Aktivierungsfunktion kann bei sehr großen Netzen zu sehr rechenintensiven Prozessen führen, da wie in der Formel 2.7 zu erkennen ist, die Exponentialfunktion oft verwendet wird.

ReLU

Die Rectified Linear Unit (ReLU) ist eine weitere weit verbreitete Aktivierungsfunktion. Bei dieser Funktion handelt es sich um eine gesättigte Funktion, da bei unendlich großen x die Ausgabe gegen unendlich geht, siehe Formel 2.9, hingegen bei Eingaben $x < 0$ die Ausgabe immer $null$ liefert, vgl. mit Formel 2.10. Diese Funktion gehört ebenfalls zu den nicht linearen Aktivierungsfunktionen. Laut [8] ist die Funktion linear, allerdings nur bei positiven Eingangsignalen.

$$\lim_{z->\infty} \sigma(x) = +\infty \tag{2.9}$$

Wie in [8] beschrieben wird diese Funktion bei Quantifizierungs-, Klassifizierungs- und Verstärkungslernproblemen verwendet. Bei Quantisierungsproblemen wird versucht die Ausgaben auf wenige diskrete Stufen zu reduzieren. Da viele Eingaben durch ReLU Funktion eine Ausgabe von $null$ erzeugen (auch bekannt als: Prinzip der geringen Wirkung [engl. Sparsity Effect]), werden auf natürlichem Wege eine Menge an Signalen eliminiert, was ein Vorteil für die Quantisierung darstellt. Durch ihr nicht lineares Verhalten ist diese Funktion auch für Klassifizierungsprobleme effizient. Häufig wird beobachtet das, bei Verstärkungsproblemen ein Rückgang des Gradienten erfolgt. Da die ReLU das *Problem des verschwindenden Gradienten* mildert, schafft sie dadurch eine stabile Lernumgebung. Zudem werden unnötige Aktionen durch das *Prinzip der geringen Wirkung* vermieden.

$$\sigma(x) = \begin{cases} x & : x > 0 \\ 0 & : x \leq 0 \end{cases} \tag{2.10}$$

$$\sigma'(x) = \begin{cases} 1 & : x > 0 \\ 0 & : x \leq 0 \end{cases} \tag{2.11}$$

Die Abbildung 2.10 zeigt die Graphen der ReLU Funktion und dessen Ableitung. Bei Eingaben kleiner als $null$ liefert die Funktion immer $null$ als Ausgabe.

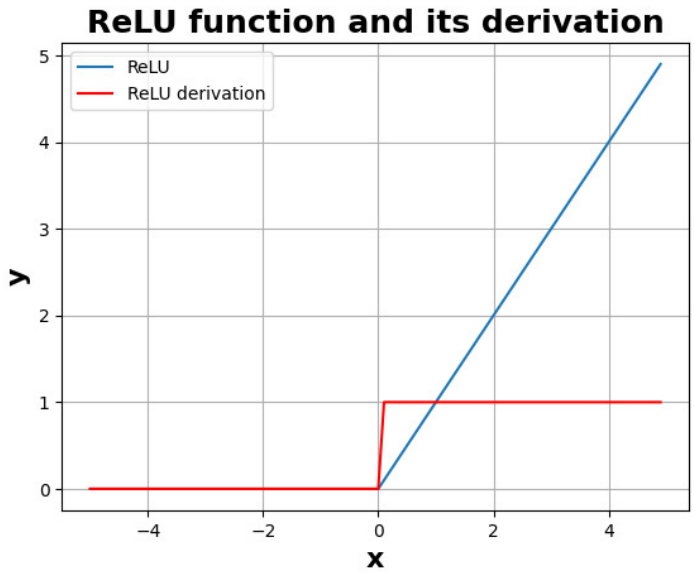


Abbildung 2.11: Graph ReLU und deren Ableitung

Die folgenden Eigenschaften sind [9] entnommen. Es wurde festgestellt, dass die ReLU-Funktion die Konvergenz des Gradientenabstiegs, aufgrund seiner linearen, nicht gesättigten Form stark beschleunigt. Da die Funktion keine Exponentialfunktion ist, sind die Kosten bei der Berechnung relativ gering. Ein Nachteil dieser Funktion ist bei sehr großen Gradienten zu sehen. Hierbei kann es passieren, das die Gewichte so aktualisiert werden, dass das Neuron nie wieder aktiviert wird und immer $null$ als Ausgabe hat. In diesem Fall können ganze Teile des Netzwerk „Tot“ bleiben.

Wie [8] entnommen kann es vorkommen, dass das Gradientenabstiegsoptimierung abstürzt. Der Grund ist, dass diese Funktion eine Diskontinuität bei $x = 0$ besitzt, was zu einem undefinierten Gradienten führt.

Leaky ReLU

Bei der Leaky ReLU handelt es sich um eine Variante der ReLU Funktion. Wie auch bei der ReLU ist die Leaky ReLU eine nicht lineare stetige Funktion, auch im Bereich von $x = 0$, was bei dem Optimierungsprozess, auch bei tiefen Netzen hilfreich sein kann. In Sachen Effizienz ist die Leaky ReLU mit der einfachen ReLU vergleichbar.

An dieser Stelle sei noch die parametrische ReLU, PreLU erwähnt. Bei der PReLU ist der Parameter α nicht fest hinterlegt, sondern kann beim Training, wie ein Bias aktualisiert werden. Die Tabelle 2.1 zeigt eine Zusammenfassung der Unterschiede und ist [10] entnommen.

	ReLU	**LReLU**	**PReLU**
Vorteil	Löst Gradientenprobleme	Löst Gradienten-probleme	Löst Gradienten-probleme
Nachteil	Sterbendes Relu-Problem	Inkonsistente Ausgabe für negative Eingabe	Feinabstimmung α
Hyperparamete	nein	nein	ja
Geschwindigkeit schnell	schneller	am schnellsten	
Genauigkeit	hoch	höher	am höchsten
Konvergenz	langsam	schnell	am schnellsten

Tabelle 2.1: Wichtige Unterschiede von ReLU Aktivierungsfunktionen

Da diese Funktion auch bei negativen x-Werten einen Gradienten ungleich $null$ besitzt, wird das „Problem der verschwindenden Gradienten“ umgangen und somit wird auch das Problem der „toten Neuronen“ verringert. Die Formel 2.13 zeigt diese Eigenschaft. Da Alpha einen sehr geringen Wert hat, kann es dennoch passieren, dass bei sehr tiefen Netzen das Lernen sehr langsam verläuft. Ein weiterer Nachteil der

Funktion sind die positiven Werte welche keinerlei Begrenzung besitzen, dieses Problem wird in Formel 2.12 deutlich. Bei extrem großen Werten kann es im Netzwerk zu nummerischen Problemen führen.

$$\sigma(x) = \begin{cases} x & : x > 0 \\ \alpha x & : x \leq 0 \end{cases} \tag{2.12}$$

$$\sigma'(x) = \begin{cases} 1 & : x > 0 \\ \alpha & : x \leq 0 \end{cases} \tag{2.13}$$

Die Abbildung 2.12 zeigt den Graphen der Leaky ReLU Funktion mit einem α-Wert von 0,2. Ebenso ist die Ableitung dieser Funktion zu sehen, als roter Graph dargestellt.

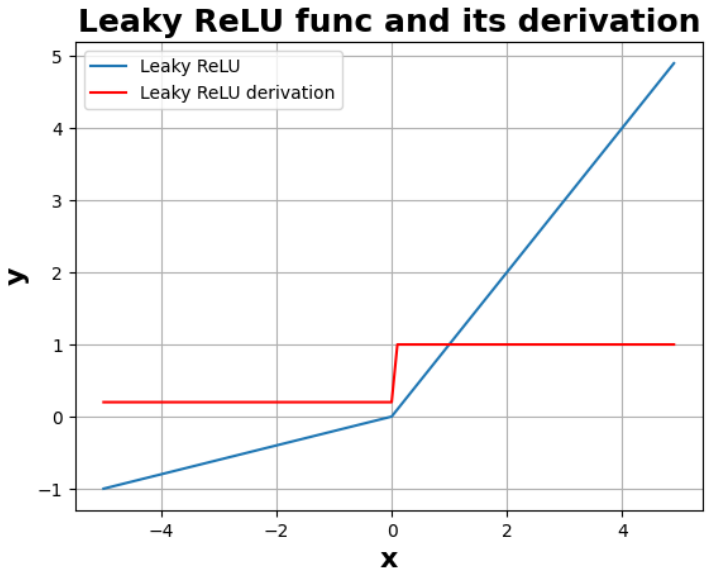


Abbildung 2.12: Graph Leaky ReLU und deren Ableitung

Der Einsatz der Leaky ReLU gegenüber der Standard ReLU, kann bei sehr tiefen neuronalen Netzen erfolgen. wenn es um sehr komplexe Klassifizierungsprobleme geht oder wenn die Standard-ReLU nicht funktioniert, beispielsweise wenn festgestellt werden kann, dass sehr viele inaktive Neuronen während des Trainings entstehen. Ein weiteres Beispiel bei der die Leaky ReLU verwendet werden sollte wird in [11] genannt. Es wird empfohlen die Leaky ReLU zu verwenden, wenn die Daten ein großes Rauschen oder viele Ausreißer aufweisen.

Gaußfunktion

Abgeleitet wird diese Aktivierungsfunktion von der Gaußschen Normalverteilung ab und symmetrisch

um den Ursprung. Somit werden für kleine positive und negative Eingabewerte, hohe Ausgabewerte und für hohe positive und negative Eingangswerte niedrige Ausgangswerte erzeugt. Daraus folgt die stärkste Aktivierung der Neuronen erfolgt bei Werten nahe dem Ursprung. Die Ausgaben liegen zwischen $null$ und $eins$. Sie ist stetig und unendlich oft differenzierbar. Diese Eigenschaft ist ein Vorteil für den Gradientenabsteig. Die Gaußfunktion ist wie Sigmoid oder Tangens Hyperbolicus sehr rechenintensiv, aufgrund Verwendung der Exponentialfunktion.

Die Gaußfunktion wird in neuronalen Netzen verwendet, die zur Mustererkennung und Klassifikation dienen. Hierbei handelt es sich um eine radiale Aktivierungsfunktion, bekannt als *Radial Basis Function* (RBF).

$$\sigma(x) = e^{-x^2} \tag{2.14}$$

$$\sigma'(x) = -2xe^{-x^2} \tag{2.15}$$

In der Abbildung 2.13 zeigt den Graphen der Gaußfunktion und dessen Ableitung. Hier ist zu erkennen, dass die Ableitung eine Antisymmetrie um den Ursprung aufweist, was mit $f'(-x) = -f'(x)$ ausgedrückt werden kann. Ebenfalls strebt die Ableitung für sehr kleine und sehr große x-Werte schnell gegen $null$, was das „Problem der verschwindenden Gradienten “ begünstigt.

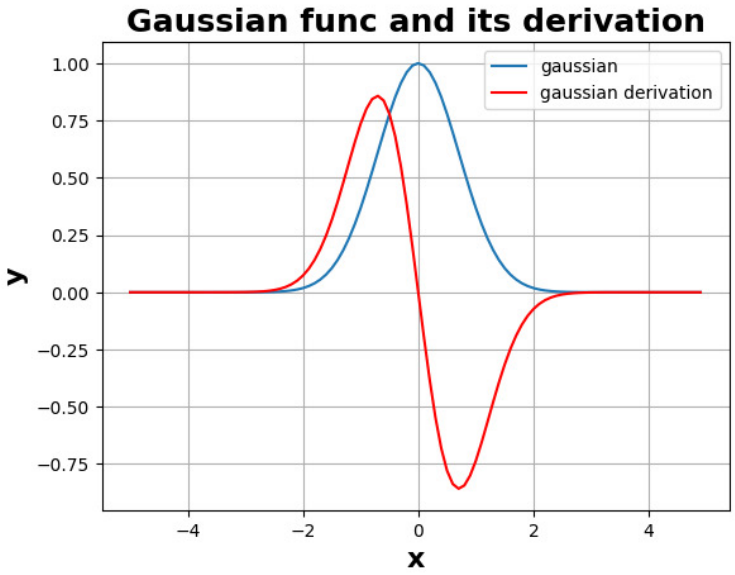


Abbildung 2.13: Graph Gaußfunktion und deren Ableitung

Ein Nachteil der Gaußfunktion kann entsteht bei sehr tiefen neuronalen Netzen. Hier kann die schnell wegen

ihrer gegen $null$ strebende Natur, zur eine Abnahme oder zum Erliegen der Aktivierungen von nachfolgenden Neuronen kommen und das Training erschweren. Die Gaußfunktion finden in den Standardnetzwerken eher wenig Anwendung. Ihre Stärken kommen erst in speziellen Netzen wie das RBF-Netz zur Geltung, wo ihre lokalen Aktivierungseigenschaften optimal genutzt werden können.

Softmax

Die Softmax Aktivierungsfunktionen wird häufig in der Ausgabeschicht eines neuronalen Netzes eingesetzt. Als Eingang dient nicht ein einzelnes Signal, wie bei den zuvor vorgestellten Funktionen, viel mehr werden alle Ausgaben der vorherigen Schicht verwendet, um Prognosen zu erstellen.

$$\sigma(x)_j = softmax(x_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ für } j = 1, ..., K. \tag{2.16}$$

Als Ausgabe wird ein Vektor erstellt der die prozentuale Verteilung der Eingangswerte darstellt. Zusammen ergeben die Ausgaben 1, also 100%, welches Formel 2.17 zeigt.

$$\sum_{i=1}^{k} softmax(z_j) = 1 \tag{2.17}$$

Die Softmax wird hauptsächlich angewandt, um Klassifizierungsprobleme die aus mehreren Klassen bestehen zu lösen. Für die eine Wahrscheinlichkeitsverteilung errechnet wird. Durch ihre Differenzierbarkeit kann der Backpropagation-Algorithmus angewandt werden, um die Gewichte zu optimieren.

Nachteilig auf die Softmax-Funktion wirken sich Ausreißer aus. Eine deutlich größeres Eingangssignal kann dominieren, auch wenn es nicht korrekt ist. Was zu einer Fehlinterpretation der Vorhersage kommen.

Maxout

Die Maxout ist ein leistungsstarkes Modell welches in [12] vorgestellte wurde und dient zur Ergänzung der Dropout-Technik. Die Funktion verwendet für die Berechnung genau wie die Softmax, die Ausgaben von Neuronen der vorherigen Schicht. Jedes Maxout-Neuron hat mehrere Transformationsvektoren und einen Bias-Vektor. Jede Transformation erhält seinen eigenen Bias-Wert. Aus den Berechnungen sucht der Algorithmus das Maximum aus den Ergebnissen heraus und liefert dieses als Output. Die Formel 2.18 zeigt den Maxout.

$$\sigma(x)_j = max(\omega_1^T x + b_1, \omega_2^T x + b_2, ..., \omega_j^T x + b_j) \tag{2.18}$$

Bei der Dropout Technik handelt es sich um ein Verfahren welches beim Training neuronaler Netze angewandt wird. Es soll das Overfitting verhindern. Durch dieses Verfahren sollen Modelle robuster und die Abhängigkeit von einzelnen Neuronen soll vermieden werden. Im Trainer kommt es zum Deaktivieren

einzelner zufällig ausgewählter Neuronen. Das Verfahren ist besonders bei tiefen Netzen mit vielen Parametern nützlich.

Arten der neuronalen Netzen

Neuronale Netze bestehen meist aus mehreren Schichten. Die Schichten wiederum enthalten eine definierte Menge an Neuronen und können in drei Arten eingeteilt werden. Die Eingabeschicht (Input layer) nimmt die Daten $\overrightarrow{x} = (x_1, ..., x_n)^T$ für das Netzwerk entgegen. In einigen Netzen werden diese Daten in dieser Schicht normalisiert. Nach der Eingabeschicht folgen ein oder mehrere versteckte Schichten (Hidden layer), welche die Daten verarbeiten. Zum Schluss kommt die Ausgabeschicht (Output layer) die die Ergebnisse $\overrightarrow{y} = (y_1, ..., y_n)^T$ des Netzwerks bereitstellt.

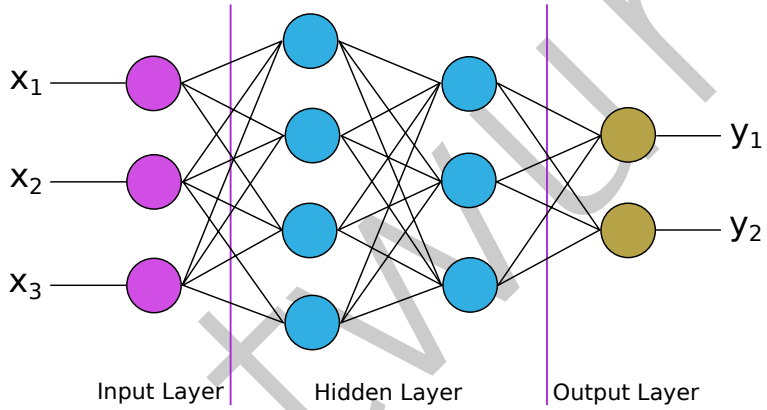


Abbildung 2.14: Aufbau eines neuronalen Netzwerkes

Alle Verbindungen oder nur ausgesuchte.

Im Folgenden werden einige neuronale Netze vorgestellt.

Einschichtige Netze (Perzeptrons)

Feedforward Neural Netzwerke (FNN) Ein FNN besteht aus einer Eingabeschicht, mindestens einer versteckten Schicht und einer Ausgabeschicht. Ein FNN ist in der Struktur und Aufbau vergleichbar mit Abbildung 2.15. Ein FNN hat in der Eingabeschicht genau so viele Neuronen wie Eingangswerte der Daten. Alle anderen Schichten können beliebig viele Neuronen enthalten. Da eine Schicht immer nur mit der nachfolgenden verbunden ist, fließen die Daten in diesem Netzwerk nur in eine Richtung. Somit können bereits verarbeitete Daten nicht noch einmal für eine weitere Verarbeitung herangezogen werden.
FNNs in vielen Bereichen eingesetzt, so beispielsweise bei der Bild- und Spracherkennung, aber auch bei Vorhersagen und Entscheidungsfindung.

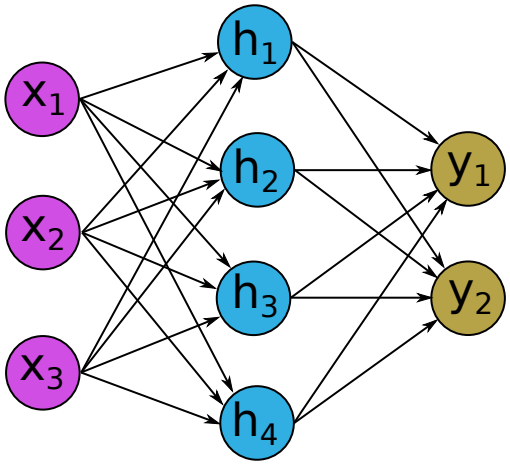


Abbildung 2.15: Aufbau eines FNN

Rekurrente neuronale Netze (RNNs)

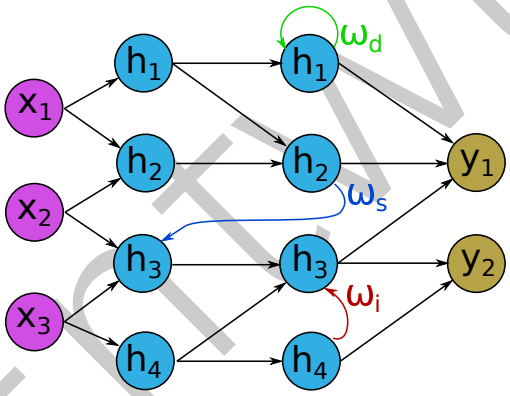


Abbildung 2.16: Aufbau eines RNN mit möglichem rückwärts gerichteten Verbindungen

Lernprozess im neuronalen Netz / Training

Ein wichtiger Bestandteil im überwachten Training für neuronale Netzwerke ist die Backpropagation. Hierbei wird die Fehlerrate, welcher durch die Vorwärtspropagation entsteht, rückwärts durch das Netz propagiert und die Gewichte werden aktualisiert. Für das Training sind Trainingsdaten notwendig, bei denen die Ergebnisse bekannt sind, welches zum Errechnen der Fehlerrate erforderlich sind. Dieser Verlust wird dann rückwärts durch das Netz propagiert. Das am häufigste angewandte Verfahren ist das Gradientenabstiegsverfahren.

Insgesamt kann der Backpropagation-Algorithmus in vier Teilschritte unterteilt werden.

Vorwärtspropagation: Zuerst werden die Trainingsdaten an das Netzwerk übergeben und die Berechnung des Ergebnisses kann erfolgen. Die Berechnung erfolgt anhand der vorgegebenen Gewichte, diese werden mit den jeweiligen Eingängen multipliziert und mit dem Bias-Wert addiert. Zum Schluss steht ein Ergebnis fest, welches am Anfang mit hoher Wahrscheinlichkeit vom erwarteten Ergebnis abweicht.

Fehlerberechnung: Das tatsächliche Ergebnis wird mit dem erwarteten Ergebnis verglichen und die Differenz gebildet. Hieraus ergibt sich der Fehler des Netzwerkes.

Rückwätspropagation: Der errechnete Fehlerwert wird zur Berechnung des Gradienten verwendet. Das Ergebnis wird dann von der Ausgangsschicht durch das Netz rückwärts propagiert und Bezug auf jedes einzelne Gewicht genommen, um die Änderung des jeweiligen Gewichtes zu errechnen.

Gewichte aktualisieren: Um die Größe der Änderung zu bestimmen wird eine Lernrate benötigt, welche die maximale Änderung der Gewichte vorgibt. Nun werden die Gewichte in entgegengesetzter Richtung des Gradienten angepasst, wodurch das Verfahren den Namen „Gradienabstieg“ erhielt.

Dieses Verfahren wiederholt, bis die *Anzahl der Epochen* erreicht ist. Die „Anzahl der Epochen “ ist ein weiterer Wert für die Backpropagation erforderlich ist. Wird Wahl der Epochen zu gering gewählt ist das Model nicht ausreichend trainiert, eine zu hohe Anzahl zieht das Training in Läge ohne maßgebliche Erfolge zu verzeichnen.

Die Backpropagation bring einige Vorteile. Sie ist einfach zu implementieren, da für fast jede Sprache eine entsprechende Library existiert. Des Weiten ist der Algorithmus flexibel und lässt sich leicht an verschiedenen Netzwerke und Architekturen anpassen und deckt somit ein weites Spektrum im Bereich KI ab. Ein weiterer Vorteil ist, es werden keine zusätzlichen Parameter benötigt, es genügen die Trainingsdaten.

Er bringt aber einige Einschränkungen mit sich. So wird ein Modell schlecht trainiert, wenn die Datenqualität ungenügend ist. Dazu zählen Rauschen, unvollständig und verzerrte Daten. Bei sehr tiefen Netzen, kann die Trainingsdauer sehr viel Zeit in Anspruch nehmen, was sich in einigen Fällen als unpraktisch erweisen kann. Durch den Matrix-basierten Ansatz steigt der Rechenbedarf mit steigender Netztiefe, was möglicherweise an die Ressourcengrenze der Hardware stößt.

2.1.6 Deep Learning

Der Bereich Deep Learning DL ist ein Bestandteil der Bereiche ML und KI. Hierbei werden tiefe neuronale Netze verwende, mit vielen Hidden Layer, um eine umfangreiche innere Struktur zu bilden. Dadurch können große Mengen an Daten effizient verarbeitet und Datenmuster erkannt werden.

Mithilfe von DL werden Problem im Bereich Klassifizierung, Regression und Vorhersagen gelöst.

> *Deep learning is a type of machine learning in which a model learns to perform classification tasks directly from images, text, or sound. Deep learning is usually implemented using a neural network architecture. The term "deep" refers to the number of layers in the network—the more layers, the deeper the network. Traditional neural networks contain only 2 or 3 layers, while deep networks can have hundreds.*
>
> MathWorks
> Introducing Deep Learning with MATLAB

In ?? wird *Depp Learning*, aufbauend auf einer *Machine Learning* Aussage beschrieben als,

> *Machine learning describes the capacity of systems to learn from problem-specific training data to automate the process of analytical model building and solve associated tasks. Deep learning is a machine learning concept based on artificial neural networks. For many applications, deep learning models outperform shallow machine learning models and traditional data analysis approaches.*
>
> Schmidhuber
> Deep Learning

2.1.7 Natural Language Processing

Natural Language Processing, kurz NLP ist ein Teilgebiet der Informatik und nutzt Deep Learning. NLP soll es digitalen Systemen in die Lage versetzen Texte und Sprachen zu erkennen, um diese zu verstehen und verarbeiten zu können. Dabei muss NLP die Bedeutung (Semantik) der Texte erkennen, die Grammatik und Beziehungen zwischen den Teilen der Sprache herstellen, Wortarten wie Verben, Adjektive und Nomen spezifizieren, sowie verschiedene Formen der Sprache beherrschen wie beispielsweise Prosa oder wissenschaftliches Schreiben.

NLP wird aber auch in anderen Bereichen eingesetzt. Mithilfe von NLP können Bilder generiert, Suchmaschinen abgefragt, Chatbots für den Kundenservice betrieben werden und Sprachassistenten wie Amazon Alexa, MS Cortana und Apple Siri nutzen ebenfalls die NLP Techniken.

Zunehmend findet NLP Einsatz im unternehmerischen Bereich. Hier werden vor allem Prozesse automatisiert um die Produktivität der Mitarbeiter zu steigern. Neben Aufgaben wie Kundensupport, Datenanalyse oder Dokumentenverwaltung kommt NLP auch in der Entwicklung von Software zum Einsatz. Hierbei werden fast

alle Segmente der Entwicklung abgedeckt, von der Codegenerierung über Test und Qualitätsmanagement bis hin zur Bereitstellung.

Erste große Erfolge haben NLP mit neuronalen Netzen wie den *Feedforward Neural Networks* und *Convolutional Neural Networks*, wie in [13] zeigt. Mit der Einführung von ChatGPT und BERT, wurde auch hier die neuen Transformer Modellen eingesetzt. NLP hat die großen Sprachmodelle erst ermöglicht.

2.2 Large Language Model

Die Teilgebiete Deep Learning und Natural Language Processing haben es den großen Sprachmodellen LLM ermöglicht kommunikationsfähig zu werden. Sie verstehen Anfragen und können Antworten generieren. Die LLMs sind in der Lage Bilder und andere Medien wie Video oder Audio zu generieren.

Diese Modelle wurden mit sehr großen Datenmengen trainiert und sind daher in der Lage natürliche Sprache zu verstehen.

2.2.1 Grundlagen

Die großen Sprachmodelle können menschliche Sprache arbeiten. Sie sind speziell für die Lösung sprachbezogene Probleme geeignet, wie Textgenerierung, Klassifizierung und Übersetzung. Sie nehmen Anfragen sog. *Prompts* entgegen und errechnen daraus die wahrscheinlichste Antwort. Des Weiteren können Prompts als Anweisung (instruction-tuning) oder in Dialogform (chat fine-tuning) gestellt werden. Die heutigen Sprachmodelle sind Modelle, welche die Transformer Technik verwenden.
Die grundlegende Funktionsweise der Large Language Models kann in vier Hauptkomponenten unterteilt werden,

1. Tokenisierung: zerlegen der Texte in einzelne Token
2. Embedding: Vergleiche mit anderen Vektoren und Einordnung in einer Gesamtstruktur
3. Vorhersage: Wahrscheinlichkeit des nächsten Tokens berechnen
4. Dekodierung: Auswahl der Ausgabestrategie

Die folgenden Unterpunkte erläutern die Hauptkomponenten näher.

Tokenisierung

Der erste Schritt besteht darin die eingegebenen Texte zu zerlegen. Die Zerlegung kann unterschiedlich gestaltet sein. Von einzelnen Zeichen, über Teilworte bis hin zu ganzen Worten. Welche Zerlegung gewählt wird, hängt maßgeblich von zwei verschiedenen Parametern ab. Zum Einem welcher Informationsgehalt ist in einem Token enthalten. Beispielsweise haben einzelne Zeichen nicht den gleichen Informationsgehalt wie ganze Worte. Der andere Aspekt ist die Anzahl der Tokens. Für ASCII bräuchte man Token, für die Worte aus der deutschen Sprache benötigt man, laut [14] etwa 300'000 bis 600'000 Token.

Der im Listing 2.1 zu sehen Methodenausschnitt einer JavaScript-Methode, wird mithilfe der Tokenisierung wie in Abbildung 2.17 gezeigt, zerlegt.

```
async function fetchUserData(apiUrl) {
  try {
    const response = await fetch(apiUrl);
    if (!response.ok) {
      throw new Error(
      'HTTP error! Status: ${response.status}'
      );
    }
    console.log('${response.status}')
    const data = await response.json();
    return data;
  } catch (error) {
    console.error("Fehler beim Abrufen:", error);
    return null;
  }
}
```

Listing 2.1: JavaScript Methode für einen API Aufruf

In der Abbildung 2.17 ist die Tokenisierung mittel Wortteilen zu erkennen. Diese Tokenisierung wurde mit https://tiktokenizer.vercel.app/ erstellt und das Model *codellama/CodeLlama-70b-hf* verwendet.

Die Tokenisierung von Quellcode stellt besondere Anforderungen. Hier muss die Sprache erkannt werden, um die Verwendung verschiedener Keywords zu realisieren. Ebenfalls müssen unterschiedliche Operatoren und Strukturen angewandt werden, z.B. === in JavaScript und :: in PHP. Bei natürlicher Sprache leite sich

```javascript
async function fetchUserData(apiUrl) {
    try {
        const response = await fetch(apiUrl);
        if (!response.ok) {
            throw new Error(
            'HTTP error! Status: ${response.status}'
            );
        }
        console.log('${response.status}')
        const data = await response.json();
        return data;
    } catch (error) {
        console.error("Fehler beim Abrufen:", error);
        return null;
    }
}
```

Abbildung 2.17: Tokenisierung einer JavaScript-Methode

die Bedeutungen der Worte durch den Kontext ab, während er bei Programmcode konsistent verarbeite werden muss. Die Semantik muss erhalten bleiben, sodass Variablennamen ihre spezifische Bedeutung behalten und die Rolle Sonderzeichen muss beachtet werden. Beispielsweise für Klammern aller Art und auch für das Semikolon. Mathematische Symbole spielen ebenso eine wichtige Rolle, wie mehrdeutige Symbole wie * (bezeichnet in C/C++ ein Malzeichen aber auch einen Zeiger), & oder |. Eine weitere Besonderheit sind die Kommentare, die für den Programmcode nicht relevant sind. Des Weiteren ist die Größe der Token zu beachten. Zu keine Token, z.B. durch Wortzerlegung kann problematisch sind, beispielsweise bei Methodennamen `is_authorized_call`, könnte in “is“, ““, “authorized“, ““ und “call“ verlegt werden. Auch sind Konstrukte wie *CamelCase* oder *Snake_case* zu beachten.

Embedding

Nach der Tokenisierung erfolgt das Embedding. D.h. die Token werden auf Vektoren abgebildet. Diese Technik ermöglicht es den neuronalen Netzen und damit den Modellen mit Text zu arbeiten. So erzeugen zwei semantisch ähnliche Token auch ähnliche Vektoren. Des Weiteren werden auch die Positionen der Token im Satz beachtet. Die Abbildung zeigt 2.18 die Wortvektoren am Beispiel des Wortes „man“, welche mithilfe der Webseite http://vectors.nlpl.eu/explore/embeddings/en/ erstellt wurde.

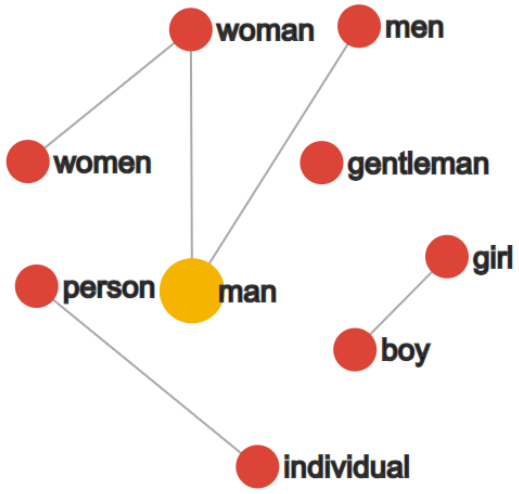


Abbildung 2.18: Tokenisierung einer JavaScript-Methode

Vorhersage

Die dritte Hauptkomponente ist die Vorhersage, ist die wichtigste der Komponenten und wird in [15] ausführlicher beschrieben. Diese Komponente macht den großen Teil der Modelle aus und ist dafür verantwortlich, dass sie zu groß sind. Die Wahl eines geeigneten Verfahrens hängt von der Fähigkeit ab mit großen Datenmengen umzugehen. Ziel ist immer möglichst viele Daten mit möglichst wenig Rechenaufwand zu trainieren. Die meisten der heutigen Modelle verwenden für die Vorhersage die Transformer-Architektur.

Dabei kommen Mechanismen wie der Selbstaufmerksamkeitsmechanismus *Self-Attention* zum Einsatz. Er verfügt über mehrere Schichten und verwendet meist Feedforward-Neuronale-Netze. Dies ermöglicht den Modellen die Eingabesequenzen parallel zu verarbeiten und die Textsequenzen in verschiedenen Kontexten zu betrachten. Die Technik verzichtet auf wiederkehrende Architektur wie die Faltung des neuronalen Netzes. Die Abbildung 2.19 zeigt alle Komponenten des Transformermodels und ist aus [15] entnommen. Hierbei wird jedes Wort mit jedem anderen Wort des Satzes in Beziehung gebracht. Dadurch lernt das Modell, in dem es die Beziehungen vergleicht und die Worte werden gewichtet.

Ein weiterer Mechanismus ist der zur Anwendung kommt, ist die *Multi-Head Attention*.

Der Tranformer liefert einen hochdimensionalen Vektor, der alle Informationen des Eingangstextes enthält.

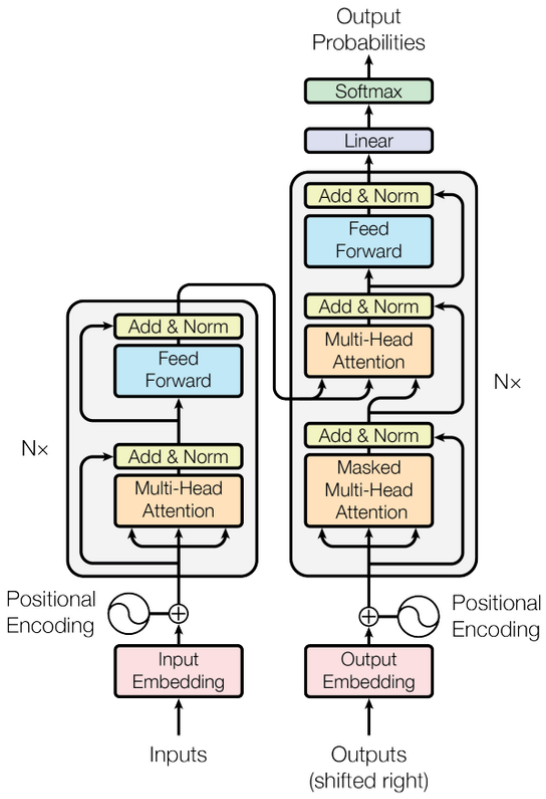


Figure 1: The Transformer - model architecture.

Abbildung 2.19: Transformermodel

Dekodierung

Dieser Vektor muss nun wieder in Text umgewandelt werden. Diese erfolgt je nach Aufgabe, um den Anforderungen gerecht zu werden. Je nach Aufgabe erfolgen dann verschiedene Verarbeitungsschritte. Beispielsweise kann bei der Textgenerierung die „Greedy Decoding“ angewandt werden.

Bei der Dekodierung geht es darum den wahrscheinlichsten oder eine Auswahl zutreffen aus n-wahrscheinlichen Token. Die Ausgaben sollen möglichst kohärente Antworten erzeugen.

2.2.2 Historie der LLM

Die Grundsteine für die Forschung der natürlichen Sprache legte Ferdinad de Saussure[2] bereits 1906 bis 1912 an der Universität von Genf. Seine Arbeit wurde von Albert Sechehaye und Charles Bally weitergeführt und veröffentlichte in seinen Namen das Werk „Grundlagen der allgemeinen Sprachwissenschaften".

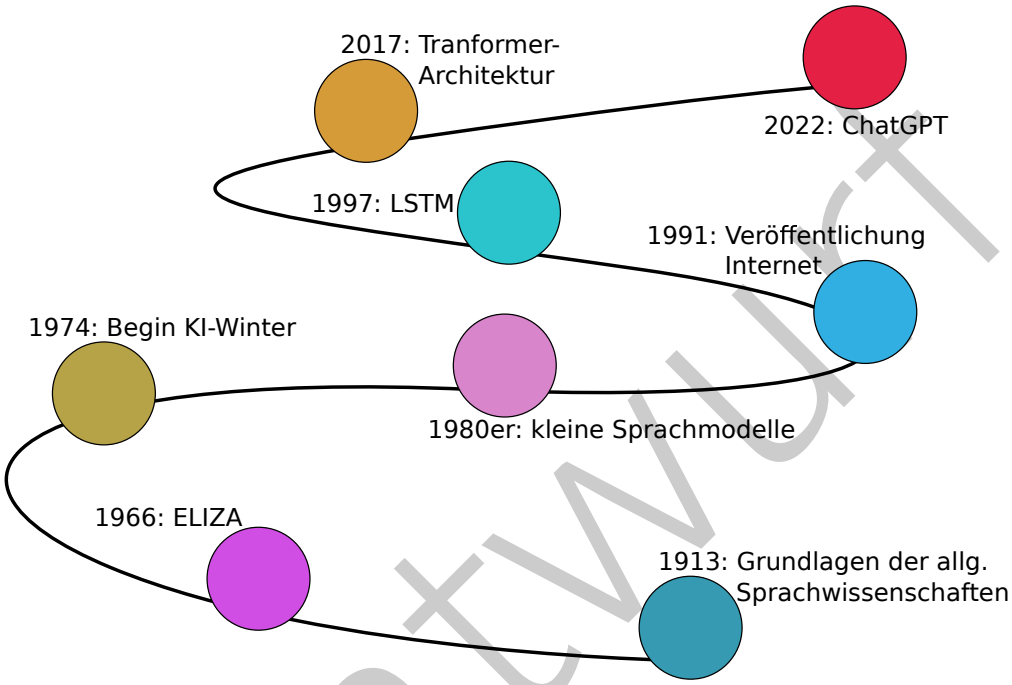


Abbildung 2.20: Entwicklung der Large Language Model

Das erste Programm welches NLP anwandte, war ELIZA. Es war der erste Chatbot, der im Jahr 1966 von MIT-Informatiker Joseph Weizenbaum entwickelt wurde. Das Programm verwendetet eine Mustererkennung in den Nutzereingaben und konnte so eine menschliche Konversation simulieren. Dies sollte der Beginn der Erforschung und Entwicklung natürlicher Sprache und noch besseren LLMs sein.

In der Zeit von 1974 bis 1980 spricht man vom „ersten KI-Winter". Viele Universitäten hatten die Forschung an neuronalen Netzen eingestellt und die Forscher organisierten sich neu und verlagerten ihren Schwerpunkt auf Wahrscheinlichkeitstheorie und Statistik. Neuronale Netze und Maschine Learning wurde hauptsächlich dafür genutzt,um Telefonanrufe zu beantworten und andere automatisierbare Prozess zu erledigen.

[2]Ferdinad de Saussure, lehrte von 1906 bis 1912 an der Universität Genf, Indogermanische und Allg. Sprachwissenschaften, sowie Sanskript. Er starb 1913

In den 1980er wurden die ersten kleinen Sprachmodelle entwickelt. Diese Modelle wurden hauptsächlich entwickelt, um das nächste Wort in einem Satz hervorsagen zu können. Sie konnten mit kleinen Datensätzen trainiert werden und berechneten nach jedem neuen Wort das nächste Wort neu.

Ende der 1980er nahm die Rechenleistung enorm zu und auch die Algorithmen bei der Verarbeitung natürlicher Sprache verbesserten sich zusehends. Hinzu kam das 1989 erfundene Internet, was 1991 der Öffentlichkeit zugänglich wurde. Damit wuchsen in den 1990er riesige Datenmengen, die für das Training der Modelle verwendet werden konnten.

Ab den 1990er Jahren wurden im Bereich Deep Learning große Fortschritte erzielt und die ersten großen Sprachmodelle wurden entwickelt. Diese Modelle arbeiteten ebenfalls mit neuronalen Netzen. Im Jahr 1997 wurden Long Short-Term Memory Netzwerke eingeführt, welche tiefe und komplexe neuronale Netzwerke ermöglichten, die große Datenmengen verarbeiten konnten.

Im Jahr 2017 führte Google Brain die Transfomer-Architektur ein, die bis heute in großen Sprachmodellen verwendet wird. Der große Hype um KI und den großen Sprachmodellen wurde mit der Einführung von ChatGPT, im November 2022 ausgelöst. In den Folgejahren haben Lösungen wie Hugging Face[3] und BARD[4] zur Weiterentwicklung erheblich beigetragen.

2.2.3 Grenzen und Probleme bei LLMs

Sind zwar toll können aber nicht alles.

Halluzinationen

Mangel an Verständnis

Kennen nicht die ganze Welt, nur ihre Muster von den Trainingsdaten. Ausgaben sind darauf beschränkt.

Ressourcenverbrauch

Hohe Rechenleistung für das Training, auch mal mehrere Monate.

Kreativitätsbegrenzung

Keine innovativen Ideen entwickeln.

[3] Hugging Face, ist ein US-Unternehmen die Anwendungen für ML entwickeln und besonders für ihre Transformer-Bibliotheken bekannt sind

[4] BARD, ist eine von Google entwickeltes kostenlosen großes Sprachmodell, welches im März 2023 veröffentlicht wurde. Es kann über Text-, Bild- oder Audiodateien interagieren

Trainingsdaten

Shit in shit out.

Transparenz

Ergebnisse schwer nachvollziehbar.

2.3 Koordinationsstrategien für LLMs

Die Large Languarge Models haben große Leistungen auf dem Gebiet der Verarbeitung natürlicher Sprache gezeigt. Zunehmend arbeiten mehrere LLMs für diese Aufgaben zusammen. In diesem Fall spricht man von Agenten, die jeweils eine LLM darstellen können.

Werden für unterschiedliche Aufgaben verschiedene Modelle verwendet, spricht man von Agenten. Ein Agent ist eine autonome Einheit. Sie ist in der Lage ihre Umwelt wahr zunehmen, Entscheidungen zu treffen und führt ihre Handlungen aus, um ein definiertes Ziel zu erreichen. Dies kann beispielsweise durch die BDI-Architektur umgesetzt werden. Jeder Agent ist auf unterschiedliche Aufgaben spezialisiert. In [16] werden Multi-Agenten-System mit Team aus der Softwareentwicklung verglichen und gleich gesetzt.

Es gibt einige Methoden Large Language Model miteinander zu kombinieren, beispielsweise „Pipeline-Architektur“ und „Modular Approaches“. Im Folgen Kapiteln werden die zwei Ansätze für die Zusammenarbeit von mehreren LLMs, *Orchestrierung* und *Multi-Agenten-System (MAS)* kurz erläutert.

2.3.1 Orchestrierung von LLMs

Bei der Orchestrierung von LLMs wird die Steuerung, der Agenten mittels eines zentralisierten Systems umgesetzt, es erfolgt eine koordinierte Nutzung. Meist wird ein Problem in Teilprobleme zerlegt und die Agenten bearbeiten Teilprobleme meist parallel. Die zentrale Steuerung entscheidet welche Teilaufgabe, welcher Agent am besten geeignet ist für die Lösung der Teilaufgabe.

Die zentrale Rolle in der Orchestrierung von LLMs übernimmt dabei der Orchestrator. Dieser steuert die Aufgabenverteilung, koordiniert und kombiniert die Ergebnisse und leitet sie in die entsprechenden Agenten oder erstellt daraus die Antwort, außerdem kann er zusätzliche Aufgaben wie Fehlerbehandlung, Skalierung, Datenschutz und Sicherheit ausführen.

Im Bereich der Softwareentwicklung mit Spezialisierung auf internetbasierte Anwendungen, bei der bestimmte Standards erwartet, spezielle Frameworks und Bibliotheken eingesetzt werden, könnte eine

Orchestrierung bei der Umsetzung der Programmcodeerstellung wie folgt beschrieben, helfen. Bei der Lösung von Anforderungen sind nicht immer alle Agent beteiligt, vielmehr sucht der Orchestrator die jeweiligen optimalen Agenten aus.

Der Orchestrator übernimmt auch hier die oben beschriebenen Aufgaben. Ein Frontend-Agent nutzt eines der großen Sprachmodelle, um Nutzeranforderungen in die Benutzeroberflächen der Anwendungen zu implementieren und könnte das Design verwalten. Gleichzeit wäre es möglich, dass dieser Agent Tools wie React.js oder Vue.js unterstützen. Für die serverseitigen Anwendungen ist der *Backend-Agent* verantwortlich und verwaltet die Logik der Anwendung. Er könnte mit Frameworks wie Node.js, Express und Django umgehen. Um die Anwendung mit einer Datenbank auszustatten, kann ein *Datenbank-Agent* eingesetzt werden. Er kennt verschiedenen Datenbanken wie MySQL oder PostgreSQL. Dieser verwaltet die Datenbank und deren Abfragen. Der *Test-Agent* testet die Anforderung die von durch den Frontend-, Backenend- oder Datenbank-Agent umsetzt wurden.

Ein letzter wichtiger Agent könnte noch der NLP-Agent sein. Dieser Agent nimmt natürliche Sprachanweisungen und Anforderungen entgegen, übersetzt diese in technische Anforderungen als Prompt für die Sprachmodelle. Die Ergebnisse der Bearbeitung werden zum Schluss von dem Agenten in eine vom Menschlichen verständliche Sprache überführt und zurückgegeben.

2.3.2 Multi-Agenten-Systeme

Multi-Agenten-Systeme (MAS) bestehen ebenfalls aus mehreren Agenten. Im Gegensatz zur Orchestrierung sind Multi-Agenten-Systeme in ihrer Steuerung dezentralisiert. Alle Agenten haben unterschiedliche Lösungsansätze für ein Problem. Je nach deren Fähigkeit hat dieser auch seine ganz eigenen Ziele, welche zu den anderen Agenten entweder als kollaborativ oder als kompetitiv ausgerichtet sind. Die Hauptarbeit zur Lösungsfindung eines Problems übernimmt der Agent, mit dem besten Lösungsansatz für das Problem. Die anderen Agenten können den ausführenden Agenten unterstützen. Um die beste Lösung zu finden, müssen die Agenten untereinander kommunizieren. Teil der Kommunikation kann es sein, einfache Informationen austauschen, um eine gemeinsame Strategie fest zulegen oder um zu Verhandeln, welcher Agent die Lösung eines Problems übernimmt.

Im Bereich der Webentwicklung mit MAS, könnte ein derartiges System wie folgt aussehen und folgende Aufgaben übernehmen. Auch hier werden nicht alle Agenten für die Lösung einer Anforderung benötigt. Vielmehr entscheidet jeder Agent für sich, ob und wie viel er zur Lösung betragen kann.
Die Abbildung 2.21 zeigt das Schema eines MAS für die Webentwicklung.

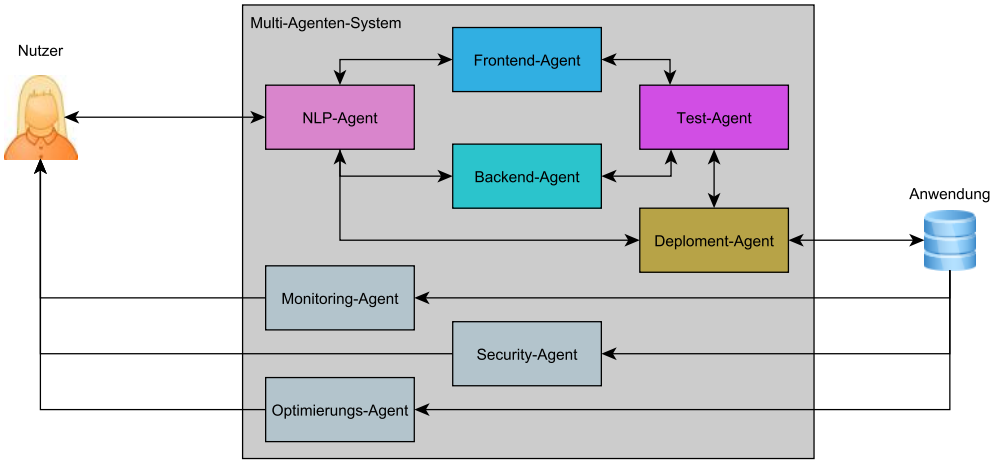


Abbildung 2.21: Multi-Agenten-System in der Webentwicklung

Ein *Frontend-Agent* ist für das Design und die Benutzeroberfläche verantwortlich. Hierbei erzeugt dieser Agent Ausgaben in HTML, JavaScript und CSS um die Oberflächen zu erstellen. Dazu kann er Frameworks, wie React verwenden und auf externe Designer Tool zugreifen. Ein weiterer Agent ist der *Backend-Agent*, der für die serverseitige Anwendung zuständig ist. Er erstellt seine Funktionen in PHP, Python oder NodeJS. Der Backend-Agent hat Zugriff auf Frameworks und externe Bibliotheken. Der erstellt und verwaltet zudem die Datenbankoperationen (CRUD-Operations). Hinzu kommt noch ein *Test-Agent*, welcher automatisierte Tests durchführt. Um die Funktionalität der Anwendung zu gewährleisten, arbeitet der Test-Agent mit dem Frontend- und Backend-Agent eng zusammen. Der Test-Agent stellt sicher, dass jegliche Codeänderung getestet wird und führt Unit-, Inetragtions- und End-to-End-Tests durch. Wird ein Fehler festgestellt, kann der Test-Agent ein Ticket erstellen oder direkt mit dem Frontend- oder Backend-Agenten kommunizieren.

Ein weiterer Agent könnte ein *Deploment-Agent* sein. Dieser führt automatische Depolyments in verschiedene Umgebungen (QA, Test oder Produktion) durch. Er ist in den Continuous Integration (CI) und Continuous Deployment (CD) Workflow integriert, welche die Bereitstellung auf verschiedenen Servern (VMware, Bare-Metal) und Cloud-Umgebungen (AWS, Azure, Google) bewerkstelligt. Des weitere könnten beispielsweise Security-Agent, Monitoring-Agent und Optimierungs-Agent Einsatz finden.

Auch hier kann ein NLP-Agent zum Einsatz kommen und die Kommunikation zwischen Mensch und System managen.

2.4 Prompt Engineering

Prompt Engineering optimiert die Antworten große Sprachmodelle, ohne Parameter, wie Bias und Gewichte des Models ändern zu müssen. Dieser Bereich hat in den letzten Jahren enorm an Bedeutung gewonnen und sich zu einer eigenen Disziplin im Bereich der Künstlichen Intelligenz entwickelt.

Ein Prompt oder Anweisung muss entweder als Anweisung oder als Frage gestellt werden. Dies kann, wie in [17] beschrieben, in Form von einer einfachen Anweisung bis hin zu detaillierten Beschreibungen oder spezifischen Aufgaben erfolgen.

[Hier Beispiel von ChatGPT oder Gemini einfügen, kann als Bild]

2.4.1 Prompt-Techniken

Siehe Prompting Techniques Hinweise für die Optimierung von Prompts. Die folgenden Techniken dienen dazu die Abfragen zu optimieren und somit eine bessere Antwort von den Sprachmodellen zu erhalten.

Zero-shot Prompting

Bei dieser Technik handelt es sich um das Senden einer einfache, klare und präzise Anweisungen, ohne Angabe von Beispielen und sonstigen zusätzlichen Informationen an die Modelle. Hierbei handelt es sich meistens um eine Domain-spezifische Anweisung. Es findet kein explizites Training vorher statt. Die Anweisung sollte ein klar definiertes Ziel haben

```
prompt_string: str = """
Generiere eine JavaScript-Funktion, die ein DIV Tag sucht und einen Text
    einfügt.
"""
```

Listing 2.2: Zero-Shot Prompt als Python-String

Im Folgenden die Antwort vom Modell. Hierbei wurde das Modell *deepseek-coder-v2* verwendet und über die API abgefragt.

```
Natürlich! Hier ist ein Beispiel für eine JavaScript-Funktion, die ein `
    DIV`-Element sucht und Text hinzufügt:

2
```javascript
function addTextToDiv(selector, text) {
    // Das Element mit dem angegebenen Selektor suchen
```
```

```
    const divElement = document.querySelector(selector);

    // Wenn das Element gefunden wurde, fügen Sie den Text hinzu
    if (divElement) {
        divElement.innerText = text;
    } else {
        console.log("Das DIV-Element mit dem angegebenen Selektor wurde
           nicht gefunden.");
    }
}

// Beispielaufruf der Funktion
addTextToDiv('div', 'Hallo, dies ist ein eingefügter Text.');
```

In diesem Code:
- `selector` ist der CSS-Selektor für das `DIV`-Element.
- `text` ist der Text, den Sie dem `DIV` hinzufügen möchten.
- Die Funktion verwendet `document.querySelector(selector)` um das Element zu finden.
- Wenn das Element gefunden wird, fügt die Funktion `innerText` des Elements den angegebenen Text hinzu.

Listing 2.3: Antwort des Zero-Shot-Prompts

Um bessere Lösungen von den Modellen zu bekommen, kann es sinnvoll sein, weitere Angaben, zum Beispiel zur Arbeitsweise oder die Definition, der zu verwendenden Bibliotheken hinzuzufügen. In [18] wird eine Methode vorgeschlagen, zur Verbesserung der Zero-shot Anweisungen.

Few-Shot Prompting

Bei komplexen Aufgaben liefert die Verwendung von Zero-shot Anweisungen oft unzureichende Ergebnisse. Hierfür finden Few-shot Prompts Verwendung. Bei dieser Technik werden ein oder mehrere Beispiele einer Antwort der Anweisung beigefügt, die als eine Art Antwortvorlage für das Modell dienen. Das Listings **??** zeigt beispielhaft den Prompt als Python-String, welcher an das Modell übertragen wurde.

```
prompt_string: str = """
Hier sind zwei Beispiele eine JavaScript Funktion die jeweils ein DIV mit
   Text füllen:
Beispiel 1
```

```
```javascript
// JavaScript Funktion, die ein DIV mit einer ID findet und Text hinzufügt
function insertTextInDiv() {
  let divElement = document.getElementById('myDiv');
  divElement.innerText = 'Hallo Welt!';
}
```
Beispiel 2
```javascript
// JavaScript Funktion, die das erste DIV-Element auf der Seite sucht und
   Text einfügt
function insertTextInFirstDiv() {
  let divElement = document.querySelector('div');
  divElement.innerText = 'Willkommen auf der Webseite!';
}

```
Aufgabe: "Generiere eine JavaScript-Funktion, die ein DIV-Tag mit einer
   bestimmten Klasse sucht und einen Text einfügt."
"""
```

Listing 2.4: Few-Shot Prompt als Python-String

Das Ergebnis vom Modell *deepseek-coder-va:latest* ist in Listing 2.5 zu sehen.

```
Hier ist eine JavaScript Funktion, die ein DIV-Tag mit einer bestimmten
   Klasse sucht und einen Text hinzufügt:

```javascript
// JavaScript Funktion, die ein DIV mit einer bestimmten Klasse findet und
    Text hinzufügt
function insertTextInDivWithClass(className) {
    let divElement = document.querySelector('div.' + className);
    if (divElement) {
        divElement.innerText = 'Hallo von der Funktion!';
    } else {
        console.log('Es wurde kein DIV mit der Klasse "' + className + '"
           gefunden.');
    }
}
```
```

```
14
Du kannst diese Funktion aufrufen und die gewünschte Klasse als Argument ü
   bergeben:

‘‘‘javascript
insertTextInDivWithClass('my-class'); // Ersetze 'my-class' durch deine
   tatsächliche Klasse
19 ‘‘‘
```

Listing 2.5: Antwort des Few-Shot-Prompts

Wie erfolgreich diese Technik ist, wird in [19] beschrieben. Wie wichtig bei der Formulierung der Anweisungen das Format und die Beschriftung ist, zeigt [20] in seiner Studie. Wird ein Beispiel angegeben, kann dazu kommen, dass das Modell nicht die richtige Antwort findet. Dann sollten mehrere Beispiele an das Modell übergeben werden.

Chain-of-Thought Prompting (CoT)

Wenn mit den Zero-shot und Few-shot Techniken nicht das gewünschte Ergebnis von den Modellen erzielt wird, könnte die Chain-of-Thought CoT Technik Verwendung finden. Bei dieser Technik wird das Modell aufgefordert, sein Vorgehen zu belegen. Mit dieser Technik kann besser nachvollzogen werden, wie im Modell der Lösungsversuch abläuft.

```
1 prompt_string: str = """
Generiere eine JavaScript-Funktion, die ein DIV-Tag sucht und einen Text
   einfügt. Denke Schritt für Schritt darüber nach, wie du die Aufgabe
   angehst.
"""
```

Listing 2.6: CoT Prompt als Python-String

Die Antwort.

```
Um eine JavaScript-Funktion zu erstellen, die ein DIV-Tag sucht und einen
   Text einfügt, können wir folgenderma ß en vorgehen:
2
1. **Auswahl des DOM-Elements**: Wir müssen zunächst das DIV-Element
   finden, dem der Text hinzugefügt werden soll. Hierfür verwenden wir die
    Methode ‘getElementById‘ oder andere Selektoren wie ‘querySelector‘.
```

```
2. **Einfügen des Textes**: Sobald wir das DIV-Element gefunden haben, kö
   nnen wir den gewünschten Text in dieses Element einfügen. Dies
   erreichen wir mithilfe der Eigenschaft ‘innerText‘ oder ‘textContent‘.

Hier ist eine detaillierte Implementierung der Funktion:

‘‘‘javascript
function insertTextIntoDiv(divId, text) {
    // 1. Auswahl des DOM-Elements
    var divElement = document.getElementById(divId);

    // Überprüfen, ob das Element existiert
    if (divElement) {
        // 2. Einfügen des Textes
        divElement.innerText = text;
    } else {
        console.error("Das DIV-Element mit der ID " + divId + " wurde
           nicht gefunden.");
    }
}
‘‘‘

### Erklärung:
1. **getElementById**: Diese Methode gibt das Element mit der angegebenen
   ID zurück. Wenn das Element nicht existiert, wird ‘null‘ zurückgegeben.
    Daher prüfen wir im Code, ob ‘divElement‘ nicht ‘null‘ ist, bevor wir
   versuchen, den Text einzufügen.
2. **innerText**: Diese Eigenschaft setzt den Text innerhalb des Elements.
    Wenn das Element bereits Inhalt enthält, wird dieser überschrieben.

### Verwendung:
Um die Funktion zu verwenden, rufen wir sie mit der ID des DIV-Elements
   und dem gewünschten Text auf:

‘‘‘html
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
```

```
    <title>Beispiel</title>
</head>
<body>
    <div id="myDiv"></div>
    <script>
        // Funktion aufrufen
        insertTextIntoDiv('myDiv', 'Hallo, Welt!');
    </script>
</body>
</html>
```
```

In diesem Beispiel wird das DIV-Element mit der ID `myDiv` gesucht und der Text "Hallo, Welt!" darin eingefügt.
```

Listing 2.7: Antwort des CoT-Prompts

Wie gut die Technik bei Modellen funktioniert, wird in [21] untersucht.

Meta Prompting

Für Meta Prompting, sind nach [22] die wichtigsten Merkmale wie folgt,

- *Strukturierung* der Prompts beispielsweise bestimmen der Denkweise oder Reihenfolge vorgeben.
- *Syntax fokussiert*, dadurch wird die Syntax als Leitvorlage für die erwartete Antwort verwendet.
- *abstrakte Beispiele* die sich mit der Struktur des Prompt befassen, nicht mit der expliziten Lösung, wie die inhaltsgesteuert Few-Shot-Prompts.
- *vielseitig* lässt es zu, das der Prompt in vielen Bereichen anwendbar ist und geben Antwort auf eine Vielzahl von Problemen, sodass der Prompt nicht jedes Mal neu geschrieben werden muss.
- *kategorischer Ansatz* strukturiert den Aufbau der Prompts in logische Anordnung und Kategorisierung

Dem Modell wird wie bei der CoT Technik angewiesen, sein Vorgehen offen zulegen. Neben dem Ergebnis, wie beim CoT, soll hierbei auch der Ablauf und die Planung der Ergebnisfindung dargestellt werden. Ein Beispiel einer Anweisung könnte folgendermaßen aussehen und wird in Listing 2.8 gezeigt.

```
prompt_string: str = """
```

```
Bevor du den Code für die Erstellung einer PHP-Funktion schreibst, die
    HTML in ein PDF speichert, erkläre die Wahl der Tools, die notwendigen
    Installationsschritte und wie du die Funktion implementierst. Verwende
    DomPDF und beschreibe den gesamten Prozess in klaren Schritten, bevor
    du den Code präsentierst.
"""
```

Listing 2.8: Meta Prompt als Python-String

Das Listing 2.9 zeigt die ausführliche Antwort. Ebenfalls wurde hier das *deepseek-code-v2:latest* Modell angewandt.

```
Natürlich! Bevor ich dir den Code für eine PHP-Funktion zur Erstellung
    eines HTML-Inhalts zu PDF mit DomPDF schreibe, möchte ich zunächst die
    Wahl der Tools und deren notwendigen Installationsschritte erklären.
    Dann werde ich den gesamten Prozess in klaren Schritten beschreiben.

### Wahl der Tools: DomPDF
DomPDF ist eine leistungsstarke PHP-Bibliothek, die es ermöglicht, HTML-
    Inhalte direkt in PDF-Dateien zu konvertieren. Es basiert auf dem
    ebenfalls beliebten TCPDF und bietet einige Vorteile wie bequeme
    Integration und eine benutzerfreundliche API.

### Notwendige Installationsschritte:
1. **Serverkonfiguration**: Stellen Sie sicher, dass Ihr Server die
    notwendigen Erweiterungen (z.B. `php-dompdf`) aktiviert hat. Dies kann
    über die `.htaccess`-Datei oder eine Konfigurationsdatei in PHP
    erfolgen.
2. **Bibliotheken herunterladen**: Laden Sie die DomPDF-Bibliothek von der
     offiziellen Website oder einem gängigen Paketmanager wie Composer
    herunter.
3. **Composer verwenden**: Wenn du Composer verwendest, füge dies in
    deiner `composer.json` Datei hinzu:
    ```json
    {
        "require": {
            "dompdf/dompdf": "^1.0"
        }
    }
    ```
    Dann führe `composer install` aus.
```

```
4. **Autoloader einbinden**: In deiner PHP-Datei, die DomPDF verwendet,
   binde den Autoloader von Composer ein:
   ```php
   require_once 'vendor/autoload.php';
   ```

Implementierung der Funktion:
1. **Initialisieren Sie DomPDF**: Erstellen Sie eine neue Instanz von
   DomPDF in Ihrer PHP-Datei.
2. **HTML-Inhalt bereitstellen**: Übergeben Sie den HTML-Code oder die URL
    des HTML-Dokuments an die DomPDF-Instanz.
3. **PDF generieren**: Rufen Sie die Methode auf, um das PDF zu erzeugen.
4. **Speichern oder Anzeigen des PDFs**: Speichern Sie das PDF in einer
   Datei oder geben Sie es direkt an den Browser aus.

Hier ist ein Beispiel für eine einfache PHP-Funktion, die HTML-Inhalt zu
   einem PDF konvertiert:

```php
<?php
require_once 'vendor/autoload.php';

use Dompdf\Dompdf;

function generatePDF($html) {
    // Instanz von Dompdf erstellen
    $dompdf = new Dompdf();

    // HTML-Inhalt setzen
    $dompdf->loadHtml($html);

    // (Optional) Konfigurieren Sie Dompdf, um die Größe des Blattes und
       andere Einstellungen anzupassen
    $dompdf->setPaper('A4', 'portrait');

    // Rendern Sie das PDF
    $dompdf->render();

    // Speichern oder Ausgeben des PDFs
```
```

```
    $output = $dompdf->output();
    file_put_contents('example.pdf', $output);
}

// Beispiel-HTML-Inhalt
$html = '<h1>Hallo, Welt!</h1><p>Dies ist ein Test für die Konvertierung
   von HTML nach PDF.</p>';
generatePDF($html);
?>
```

```

### Zusammenfassung:
1. **Installiere DomPDF und andere benötigte Bibliotheken**.
2. **Initialisiere eine Instanz von DomPDF** in deiner PHP-Datei.
3. **Setze den HTML-Inhalt oder die URL des HTML-Dokuments**.
4. **Rendern Sie das PDF** und speichern oder geben Sie es aus.

Mit diesem Prozess kannst du einfach HTML-Inhalte in professionelle PDFs
   umwandeln, die dann gespeichert oder weiter verwendet werden können.
```

Listing 2.9: Antwort des Meta-Prompts

Meta-Prompts sind Token-Effizient und verringern die benötigte Anzahl an Token, da der Schwerpunkt wie beschrieben auf der Struktur liegt, nicht auf den expliziten Inhalt.

Prompt Chaining

Hierbei wird eine komplexe Aufgabe in Unteraufgaben zerlegt. Die Antwort einer Unteraufgabe dient als Eingabe für die nächste Unteraufgabe. Diese Zerlegung ist hilfreich, um Komplexität einer Aufgabe zu verringern und eine Überforderung der Modelle zu verhindern. Durch diese Technik ist eine schrittweise Näherung an die Gesamtlösung der Aufgabe möglich.

Im Beispiel soll das Sprachmodell wieder eine PHP Funktion schreiben, die eine HTML Zeichenkette als PDF speichert.

```
prompt_string: str = """
Nenne mögliche PHP Bibliotheken, mit der PDF Dateien erstellt werden kö
   nnen.
"""
```

Listing 2.10: Chain Prompt Nr. 1 als Python-String

```
 Es gibt mehrere PHP Bibliotheken, die es ermöglichen, PDF-Dateien zu
    erstellen. Hier sind einige beliebte und weit verbreitete Bibliotheken
    :

1. **TCPDF**:
   - TCPDF ist eine leistungsstarke Open-Source-Bibliothek zur Erstellung
     von PDF-Dokumenten in PHP. Es bietet umfassende Unterstützung für
     verschiedene Funktionen wie Texte, Grafiken, Tabellen und mehr.
   - [Offizielle Website](https://tcpdf.org/)
   - [GitHub-Repository](https://github.com/tecnickcom/tc-lib-pdf)

2. **MPDF**:
   - MPDF ist eine weitere leistungsstarke PHP Bibliothek zur PDF-
     Generierung, die sehr beliebt ist und viele Funktionen bietet,
     einschließlich der Unterstützung für Unicode-Zeichen.
   - [Offizielle Website](https://mpdf.github.io/)
   - [GitHub-Repository](https://github.com/mpdf/mpdf)

3. **FPDF**:
   - FPDF ist eine einfache und leichtgewichtige Bibliothek zur PDF-
     Generierung in PHP, die auf der FPDI-Bibliothek basiert. Es ermö
     glicht das Erstellen von PDF-Dokumenten mit Text, Grafiken und
     vielen anderen Elementen.
   - [Offizielle Website](https://tcpdf.org/downloads/)
   - [GitHub-Repository](https://github.com/fpdf/fpdf)

4. **DomPDF**:
   - DomPDF ist eine PHP-Implementierung von PDFLib, einer
     leistungsstarken Bibliothek zur Erstellung von PDF-Dokumenten in C
     ++. Es bietet auch die Möglichkeit, HTML in PDF umzuwandeln.
   - [Offizielle Website](https://dompdf.github.io/)
   - [GitHub-Repository](https://github.com/dompdf/dompdf)

5. **SnappyPdf**:
   - SnappyPdf ist eine PHP Bibliothek, die auf der wunderbaren Pdfcrowd
     API basiert und ermöglicht das Erstellen von PDF-Dokumenten aus HTML
      oder Textinhalten.
   - [Offizielle Website](https://github.com/KnpLabs/snappy)
   - [GitHub-Repository](https://github.com/KnpLabs/snappy)
```

```
27
Diese Bibliotheken bieten verschiedene Funktionen und Flexibilität, sodass
    Sie je nach Anwendungsfall die passende Wahl treffen können.
```

Listing 2.11: Antwort des Chain-1-Prompts

```
prompt_string: str = """
2 Welche der folgenden Bibliotheken bieten die beste Unterstützung der
    Entwickler,
* TCPDF
* MPDF
* FPDF
* DomPDF
7 * SnappyPdf
"""
```

Listing 2.12: Chain Prompt Nr. 2 als Python-String

```
Um zu beurteilen, welche der angegebenen Bibliotheken die beste Unterstü
    tzung für Entwickler bietet, ist es wichtig, verschiedene Kriterien zu
    berücksichtigen. Diese können unter anderem die Funktionalität, die
    Aktualität der Dokumentation, das Umfeld der Community, die Unterstü
    tzung von Drittanbietern und die allgemeine Benutzerfreundlichkeit
    umfassen.
2

Hier ist eine kurze Bewertung einiger dieser Bibliotheken:

1. **TCPDF**: TCPDF (HTML2FPDF) ist eine leistungsstarke PDF-Bibliothek,
    die hauptsächlich in PHP entwickelt wurde und auf HTML2FPDF basiert. Es
     bietet eine Vielzahl von Funktionen für das Erstellen und Bearbeiten
    von PDF-Dokumenten. Die Dokumentation ist recht ausführlich, und es
    gibt eine aktive Community.

7 2. **MPDF**: MPDF (Multi-Purpose PDF Library) ist ebenfalls in PHP
    geschrieben und bietet ähnliche Funktionen wie TCPDF. Es unterstützt
    HTML sowohl im Text- als auch im Modus, was es für Entwickler attraktiv
     macht, die mit Webtechnologien arbeiten. Die Dokumentation ist gut und
     die Community ist aktiv.

3. **FPDF**: FPDF (Free Pascal PDF) ist eine andere weit verbreitete PHP-
    Bibliothek zum Erstellen von PDF-Dokumenten. Es ist in Free Pascal
```

```
   geschrieben, aber auch viele PHP-Entwickler bevorzugen es für seine
   einfache Verwendung und die Fähigkeit, HTML zu rendern. Die
   Dokumentation ist gut verständlich.

4. **DomPDF**: DomPDF ist eine portierte Version von MPDF in PHP, die auf
   der DOM-Basis arbeitet. Es unterstützt das Rendern von HTML und bietet
   eine hohe Flexibilität bei der Erstellung komplexer PDF-Dokumente. Die
   Dokumentation ist gut, aber die Community ist kleiner als bei anderen
   Bibliotheken.
12
5. **SnappyPdf**: SnappyPdf verwendet die C-Bibliothek "snappy" von Google
    für das Komprimieren und Dekomprimieren der PDF-Dateien. Es unterstü
   tzt viele verschiedene Backends wie Ghostscript, Qpdf oder MuPDF. Die
   Verwendung erfordert jedoch evtl. zusätzliche Abhängigkeiten und ist
   nicht in allen Sprachen verfügbar.

**Empfehlung**: Wenn Sie eine Bibliothek für PHP-Projekte suchen, bei
   denen HTML unterstützt wird und Sie eine reaktionsfähige Unterstützung
   der Entwickler sowie eine aktive Community benötigen, würde ich **TCPDF
   ** oder **MPDF** empfehlen. Beide Bibliotheken sind gut dokumentiert
   und haben aktive Communities. FPDF ist ebenfalls ein guter Ersatz, vor
   allem wegen seiner guten Leistung und der größeren Verbreitung im PHP-
   Umfeld.

17 **SnappyPdf** sollte nur dann in Betracht gezogen werden, wenn Sie
   speziell eine plattformübergreifende Lösung benötigen und bereits über
   die notwendigen Abhängigkeiten verfügen.
```

Listing 2.13: Antwort des Chain-2-Prompts

```
prompt_string: str = """
Erstelle eine PHP Funktion, die eine HTML Zeichenkette in einer PDF Datei
   speichert. Dafür soll die PHP Bibliothek ‘DomPDF‘ verwendet werden.
3 """
```

Listing 2.14: Chain Prompt Nr. 3 als Python-String

```
Um eine PHP Funktion zu erstellen, die eine HTML-Zeichenkette in einer PDF
   -Datei speichert, kann die PHP Bibliothek ‘DomPDF‘ verwendet werden.
   Hier ist ein Beispiel, wie Sie dies tun können:
```

````
2
1. Installieren Sie zuerst die DomPDF Bibliothek über Composer:
```bash
composer require dompdf/dompdf
```
7
2. Erstellen Sie eine PHP Funktion, die die HTML-Zeichenkette enthält und
   die PDF speichert:

```php
<?php
12 require_once __DIR__ . '/vendor/autoload.php';

use Dompdf\Dompdf;

function saveHtmlToPdf($html) {
17     // Erstellen Sie eine Instanz von Dompdf
    $dompdf = new Dompdf();

    // Setzen Sie die HTML-Inhalte in Dompdf
    $dompdf->loadHtml($html);
22
    // (Optional) Konfigurieren Sie die PDF Optionen
    $dompdf->setPaper('A4', 'portrait');

    // Rendern Sie das PDF
27     $dompdf->render();

    // Speichern Sie das PDF in einer Datei
    $output = $dompdf->output();
    file_put_contents('output.pdf', $output);
32 }

// Beispiel HTML-Inhalt
$html = '<!DOCTYPE html>
<html lang="de">
37 <head>
    <meta charset="UTF-8">
    <title>Beispielseite</title>
````
````

```
</head>
<body>
    <h1>Hallo Welt!</h1>
    <p>Dies ist ein Beispiel für die Verwendung von DomPDF in PHP.</p>
</body>
</html>';

// Rufen Sie die Funktion auf, um das HTML als PDF zu speichern
saveHtmlToPdf($html);
?>
```

```

In diesem Beispiel:
1. Wir laden zuerst die DomPDF Bibliothek über Composer.
2. Eine PHP Funktion `saveHtmlToPdf` wird definiert, die eine HTML-
   Zeichenkette als Parameter enthält.
3. Innerhalb der Funktion erstellen wir eine Instanz von Dompdf und setzen
    den in der Funktion übergebenen HTML-Inhalt.
4. Wir rendern das PDF und speichern es als `output.pdf`.

Sie können diese Funktion aufrufen und die gewünschte HTML-Zeichenkette an
    sie übergeben, um eine PDF-Datei zu erstellen und zu speichern.
```

Listing 2.15: Antwort des Chain-3-Prompts

Tree of Thoughts (ToT)

Diese Technik wurde von [23] und [24] vorgeschlagen. (ToT) kommt bei komplexen Anforderungen zum Einsatz, wenn einfache Techniken, die zuvor genannt wurden, nicht mehr ausreichen. Auch bei dieser Technik wird die Anforderung in keine Aufgaben zerlegt. Dann werden mehrere Lösungen pro Aufgabe erstellt und im Anschluss bewertet. Dabei entsteht eine Baustruktur, von der die besten Lösungen ausgesucht werden.
Im Folgenden ein Beispiel einer Teilaufgabe, bei der drei mögliche Lösungsvorschläge vom Modell erstellt wurden. Die Listings 2.16, 2.17 und 2.18 zeigen die Ergebnisse des Modells. Es wurden jeweils andere PHP Bibliotheken für die Lösung vorgeschlagen. Als Modell wurde *deepseek-coder-v2* verwendet. Als Nutzereingabe wurde folgendes an das Modell übergeben,

> *Erstelle drei verschiedene Methoden in PHP, die eine HTML Zeichenkette in ein PDF umwandeln und es als Datei, mit angegebenen Namen speichern.*

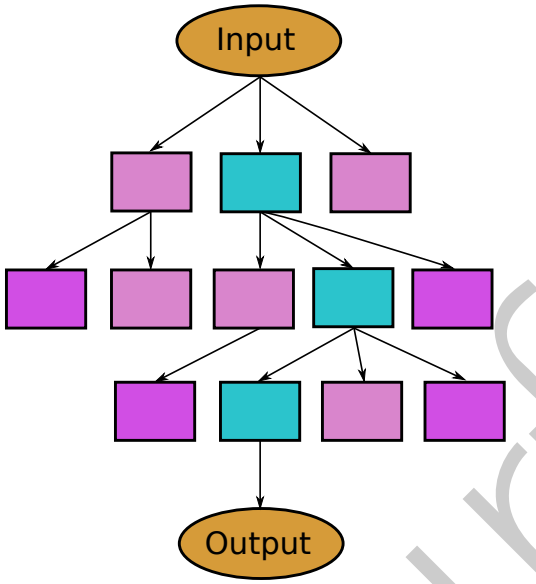


Abbildung 2.22: Baumstruktur der „Tree of Thoughts“ Technik

Die erste Lösung beinhaltet die Bibliothek *Dompdf*.

```php
<?php
require 'vendor/autoload.php';

use Dompdf\Dompdf;

function createPDF($html, $filename)
{
  $dompdf = new Dompdf();
  $dompdf->loadHtml($html);
  $dompdf->setPaper('A4', 'portrait');
  $dompdf->render();
  file_put_contents($filename, $dompdf->output());
}

// ... more.
```

Listing 2.16: Ausgabe für DOMPDF Bibliothek

```php
<?php
require_once('mpdf/autoload.inc.php');
```

```
use Mpdf\Mpdf;

function createPDF($html, $filename)
{
  $mpdf = new Mpdf();
  $mpdf->WriteHTML($html);
  $mpdf->Output($filename, 'F');
}

// ... more.
```

Listing 2.17: Ausgabe für MPDF Bibliothek

Als dritte Ausgabe liefert coder,

```
<?php
require_once('tcpdf/tcpdf.php');

class CustomPDF extends TCPDF
{
  public function Header()
  {
    $this->SetFont('helvetica', 'B', 20);
    $this->Cell(0, 15, 'Beispiel-PDF', 0, false, 'C', 0, '', 0, false, 'M'
        , 'M');
  }
}

function createPDF($html, $filename) {
  $pdf = new CustomPDF(PDF_PAGE_ORIENTATION, PDF_UNIT, PDF_PAGE_FORMAT,
     true, 'UTF-8', false);
  $pdf->SetDefaultMonospacedFont(PDF_FONT_MONOSPACED);
  $pdf->SetMargins(PDF_MARGIN_LEFT, PDF_MARGIN_TOP, PDF_MARGIN_RIGHT);
  $pdf->SetHeaderMargin(PDF_MARGIN_HEADER);
  $pdf->SetFooterMargin(PDF_MARGIN_FOOTER);
  $pdf->setPrintHeader(false);
  $pdf->setPrintFooter(false);
  $pdf->AddPage();
  $pdf->writeHTML($html, true, false, true, false, '');
  $pdf->Output($filename, 'F');
```

```
}

// ... more.
```

Listing 2.18: Ausgabe für TCPDF Bibliothek

2.4.2 Grenzen beim Prompt-Engineering für LLMs

Trotz der bemerkenswerten linguistischen Leistung, stoßen große Sprachmodelle an ihre Grenzen, unter anderem wie in [17] beschrieben,

2.5 Grundlagen bei der Entwicklung von Webanwendungen

Webanwendung

Stand der Forschung

3

Mit der Einführung von ChatGPT für die breite Öffentlichkeit, am 30. November 2022, wurde ein Hype um die großen Sprachmodelle ausgelöst, der von da an als treiben Kraft, hinter ihrer Entwicklung gesehen werden kann. In einigen Artikel ist sogar die Rede von einer,

> *Zeitenwende, wie sie nur alle 30-40 Jahre erleben und wie wir sie zuletzt mit der Einführung des World Wide Web gesehen haben*
>
> Siegfried Handschuh, entnommen aus [25]
> Grosse Sprachmodelle

Wo knüpfe ich an?

3.1 Methoden und Ansätze

3.2 Forschungslücken und zukünftige Forschung

Künftige Forschung.

3.2.1 Identifikation von Forschungslücken

3.2.2 Zukünftige Forschungsrichtungen

Methodik

4

4.1 Auswahl der LLM

4.2 Prompt-Engineering

Prompt-Engineering mit verschiedenen Ansätzen, die zu testen sind.

Implementierung

5

5.1 Modelle lokal aufsetzen

Als Server dient ein Debian 12 System.

5.1.1 Install Ollama

Ein Skript ausführen

```
curl -fsSL https://ollama.com/install.sh | sh
```

Ein Model laden und im Anschluss starten, Beispiel

```
ollama pull deepseek-coder-v2:16b \
ollama run deepseek-coder-v2:16b
```

Config Ollama
Set correct IP and Post in /etc/systemd/system/ollama.service

```
  diff --git a/ollama.service b/ollama.service
  --- a/ollama.service
  +++ b/ollama.service
4 @@ -10,3 +10,4 @@
  RestartSec=3
  Environment="PATH=/usr/local/bin:/usr/bin"
-
+   Environment="OLLAMA_HOST=0.0.0.0"
9 +   Environment="OLLAMA_MODELS=/home/ai/models"
+
```

Listing 5.1: Ollama Hostanpasssng für Netzwerkbetrieb

5.1.2 Open WebUI

Optional kann ein grafisches Tool, zum Testen und verwalten vom Ollama-Server im Netzwerk installiert werden. Der Aufruf der UI, kann mittel Browser erfolgen. Hier wird die IP und der Port 8080 angegeben. Beispiel `http://192.168.2.45:8080`.

```
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
-o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io \
docker-buildx-plugin docker-compose-plugin
docker run -d --network=host -v open-webui:/app/backend/data \
-e OLLAMA_BASE_URL=http://127.0.0.1:11434 --name open-webui \
--restart always ghcr.io/open-webui/open-webui:main
```

5.1.3 Python Client

```
pip3 install langchain
pip3 install ollama
pip3 install mistral
```

Evaluation

6

6.1 Einfache HTML Seite

6.1.1 ChatGPT 3.5

Python Funktion

Schreibe eine Python Funktion die FPDF Dokument als Eingabe erhält und eine Zeichenkette zurückliefert.
(engl.: Write a Python function that takes FPDF document as input and returns a string.)

Antwort ChatGPT

Antwort.

Copilot

Klar, ich kann dir dabei helfen! Hier ist ein Beispiel für eine Python-Funktion, die ein FPDF-Dokument als Eingabe erhält und den Textinhalt als Zeichenkette zurückliefert. Dafür verwenden wir die Bibliothek PyMuPDF (auch bekannt als fitz), um das PDF zu lesen:

```
import fitz  # PyMuPDF

def pdf_to_string(pdf_path):
    # Öffne das PDF-Dokument
    document = fitz.open(pdf_path)
    text = ""

    # Iteriere durch jede Seite und extrahiere den Text
    for page_num in range(len(document)):
        page = document.load_page(page_num)
        text += page.get_text()

    return text

# Beispielverwendung
pdf_path = "dein_dokument.pdf"
text_inhalt = pdf_to_string(pdf_path)
print(text_inhalt)
```

Stelle sicher, dass du die PyMuPDF-Bibliothek installiert hast. Du kannst sie mit folgendem Befehl installieren:

```
pip install pymupdf
```

Diese Funktion öffnet das PDF-Dokument, iteriert durch jede Seite und extrahiert den Text. Wenn du Fragen hast oder weitere Anpassungen benötigst, lass es mich wissen!

Python Funktion

Schreibe eine Python Funktion die FPDF Dokument als Eingabe erhält und eine Zeichenkette zurückliefert. Verwende dafür die Python Bibliothek FPDF.
(engl.: Write a Python function that takes FPDF document as input and returns a string.)

Copilot

Die Bibliothek FPDF wird hauptsächlich zum Erstellen von PDF-Dokumenten verwendet und bietet keine direkte Methode zum Extrahieren von Text aus einem PDF. Um Text aus einem PDF-Dokument zu extrahieren, könntest du eine andere Bibliothek wie PyMuPDF oder PyPDF2 verwenden.
Falls du jedoch darauf bestehst, FPDF zu verwenden, um ein PDF zu erstellen und dann den Text aus diesem PDF zu extrahieren, könntest du den Text während des Erstellungsprozesses speichern und später zurückgeben. Hier ist ein Beispiel, wie du das machen könntest:

```
from fpdf import FPDF

class PDF(FPDF):
    def __init__(self):
        super().__init__()
        self.text_content = ""

    def add_text(self, text):
        self.text_content += text + "\n"
        self.set_xy(10, 10)
        self.set_font('Arial', 'B', 12)
        self.multi_cell(0, 10, text)

    def get_text(self):
        return self.text_content

    def create_pdf_with_text(text):
        pdf = PDF()
            pdf.add_page()
        pdf.add_text(text)
        pdf.output("output.pdf")
        return pdf.get_text()

# Beispielverwendung
text = "Dies ist ein Beispieltext für das PDF."
text_inhalt = create_pdf_with_text(text)
print(text_inhalt)
```


In diesem Beispiel wird der Text während des Erstellungsprozesses des PDFs in einer Zeichenkette gespeichert und kann später abgerufen werden. Wenn du Fragen hast oder weitere Anpassungen benötigst, lass es mich wissen!

Python Funktion

Schreibe eine Python Funktion die FPDF Dokument als Eingabe erhält und eine Zeichenkette zurückliefert. Verwende dafür die Python Bibliothek FPDF und speicher das Ergebnis nicht als Datei zwischendurch ab.

Copilot

Die Bibliothek FPDF ist hauptsächlich zum Erstellen von PDF-Dokumenten gedacht und bietet keine direkte Methode zum Extrahieren von Text aus einem PDF. Um Text aus einem PDF-Dokument zu extrahieren, könntest du eine andere Bibliothek wie PyMuPDF oder PyPDF2 verwenden.
Falls du jedoch darauf bestehst, FPDF zu verwenden, um ein PDF zu erstellen und dann den Text aus diesem PDF zu extrahieren, könntest du den Text während des Erstellungsprozesses speichern und später zurückgeben. Hier ist ein Beispiel, wie du das machen könntest:

```
from fpdf import FPDF

class PDF(FPDF):
    def __init__(self):
        super().__init__()
        self.text_content = ""

    def add_text(self, text):
        self.text_content += text + "\n"
        self.set_xy(10, 10)
        self.set_font('Arial', 'B', 12)
        self.multi_cell(0, 10, text)

    def get_text(self):
        return self.text_content

    def create_pdf_with_text(text):
        pdf = PDF()
        pdf.add_page()
        pdf.add_text(text)
        return pdf.get_text()

# Beispielverwendung
text = "Dies ist ein Beispieltext für das PDF."
text_inhalt = create_pdf_with_text(text)
print(text_inhalt)
```

In diesem Beispiel wird der Text während des Erstellungsprozesses des PDFs in einer Zeichenkette gespeichert und kann später abgerufen werden. Wenn du Fragen hast oder weitere Anpassungen benötigst, lass es mich wissen!

6.2 Einfache HTML Seite

6.2.1 ChatGPT 3.5

HTML Startseite

Schreibe eine einfache HTML Seite für die Startseite eines Blogs.

ChatGPT 3.5

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mein Blog</title>
  <style>
    // styles.
  </style>
</head>
<body>
  <header>
    <h1>Willkommen auf meinem Blog</h1>
  </header>
  <nav>
    <a href="#">Startseite</a>
    <a href="#">Über mich</a>
    <a href="#">Blog</a>
    <a href="#">Kontakt</a>
  </nav>
  <div class="container">
    <article>
      <h2>Erster Blogbeitrag</h2>
      <p>Willkommen zu meinem ersten Blogbeitrag!</p>
    </article>
  <footer>
    <p>&copy; 2024 Mein Blog. Alle Rechte vorbehalten.</p>
  </footer>
</body>
</html>
```

ChatGPT 3.5

```
<!DOCTYPE html>
<html lang="de">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mein Blog</title>
<style>
// styles.
</style>
</head>
<body>
<header>
<h1>Willkommen auf meinem Blog</h1>
</header>
<nav>
<a href="#">Startseite</a>
<a href="#">Über mich</a>
<a href="#">Blog</a>
<a href="#">Kontakt</a>
</nav>
<div class="container">
<article>
<h2>Erster Blogbeitrag</h2>
<p>Willkommen zu meinem ersten Blogbeitrag!</p>
</article>
<footer>
<p>&copy; 2024 Mein Blog. Alle Rechte vorbehalten.</p>
</footer>
</body>
</html>
```

Anwendungsszenarien

7

Diskussion und Ausblick

8

Wie in [26] beschrieben,

Fazit

9

Literatur

[1] Volker M. Banholzer. *Künstliche Intelligenz als Treiber der Veränderung in der Unternehmenskommunikation 4.0?* Bd. 1/2020. Technische Hochschule Nürnberg Georg-Simon-Ohm, 2020. URL: https://www.th-nuernberg.de/fileadmin/fakultaeten/amp/amp_docs/K%C3%BCnstliche_Intelligenz_und_die_Rolle_n__von_Unternehmenskommunikation_Banholzer_IKOM_WP_1_2020__fin-1.pdf.

[2] *Digitale Transformation: Fallbeispiele und Branchenanalysen*. 2022. URL: https://library.oapen.org/bitstream/handle/20.500.12657/57358/978-3-658-37571-3.pdf?sequence=1&utm_source=textcortex&utm_medium=zenochat#page=70 (besucht am 19. 10. 2024).

[3] Erin Yepis. *Developers want more, more, more: the 2024 results from Stack Overflow's Annual Developer Survey*. 24. Juli 2024. URL: https://stackoverflow.blog/2024/07/24/developers-want-more-more-more-the-2024-results-from-stack-overflow-s-annual-developer-survey/ (besucht am 09. 08. 2024).

[4] Pekka Ala-Pietilä u. a. *Eine Definition der KI: Wichtigste Fähigkeiten und Wissenschaftsgebiete*. 5. März 2019. URL: https://elektro.at/wp-content/uploads/2019/10/EU_Definition-KI.pdf (besucht am 10. 09. 2024).

[5] Johanna Pahl. *Zeichnung einer biologische Zelle*. 26. Sep. 2024.

[6] Jason Brownlee. *How to Choose an Activation Function for Deep Learning*. 22. Jan. 2021. URL: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (besucht am 19. 09. 2024).

[7] Siddharth Sharma u. a. *Activation Functions in Neural Networks*. Issue 12. Apr. 2020, S. 310–316. URL: https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf (besucht am 19. 09. 2024).

[8] Brosnan Yuen u. a. „Universal activation function for machine learning“. In: *Scientific Reports* 11.1 (21. Sep. 2021), S. 18757. ISSN: 2045-2322. DOI: 10.1038/s41598-021-96723-8. URL: https://doi.org/10.1038/s41598-021-96723-8.

[9] *CS231n Convolutional Neural Networks for Visual Recognition*. URL: https://cs231n.github.io/neural-networks-1/ (besucht am 20. 09. 2024).

[10] Nikhil Bhargav. *ReLU vs. LeakyReLU vs. PReLU | Baeldung on Computer Science*. 27. Nov. 2023. URL: https://www.baeldung.com/cs/relu-vs-leakyrelu-vs-prelu (besucht am 20. 09. 2024).

[11] Srikari Rallabandi. „Activation functions: ReLU vs. Leaky ReLU - Srikari Rallabandi - Medium“. In: (27. März 2023). URL: https://medium.com/@sreeku.ralla/activation-functions-relu-vs-leaky-relu-b8272dc0b1be.

[12] Ian J. Goodfellow u. a. *Maxout Networks*. version: 4. 20. Sep. 2013. DOI: 10.48550/arXiv.1302.4389. arXiv: 1302.4389[cs,stat]. URL: http://arxiv.org/abs/1302.4389 (besucht am 20. 09. 2024).

[13] Yoav Goldberg. „A Primer on Neural Network Models for Natural Language Processing“. In: *Journal of Artificial Intelligence Research* 57 (20. Nov. 2016), S. 345–420. DOI: 10.1613/jair.4992. URL: https://jair.org/index.php/jair/article/view/11030.

[14] *Der Umfang des deutschen Wortschatzes*. 2020. URL: https://www.duden.de/sprachwissen/sprachratgeber/Zum-Umfang-des-deutschen-Wortschatzes (besucht am 23. 09. 2024).

[15] Ashish Vaswani u. a. *Attention is all you need*. 12. Juni 2017. URL: https://arxiv.org/abs/1706.03762 (besucht am 23. 09. 2024).

[16] Zhuoyun Du u. a. *Multi-Agent Software Development through Cross-Team Collaboration*. 13. Juni 2024. URL: https://arxiv.org/abs/2406.08979 (besucht am 04. 10. 2024).

[17] Xavier Amatriain. *Prompt Design and Engineering: Introduction and Advanced Methods*. 24. Jan. 2024. URL: https://arxiv.org/abs/2401.14423v3 (besucht am 12. 10. 2024).

[18] Jason Wei u. a. *Finetuned language models are Zero-Shot learners*. 3. Sep. 2021. URL: https://arxiv.org/abs/2109.01652 (besucht am 12. 10. 2024).

[19] Tom B. Brown u. a. *Language Models are Few-Shot Learners*. 28. Mai 2020. URL: https://arxiv.org/abs/2005.14165 (besucht am 12. 10. 2024).

[20] Sewon Min u. a. *Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?* 25. Feb. 2022. URL: https://arxiv.org/abs/2202.12837 (besucht am 12. 10. 2024).

[21] Jason Wei u. a. *Chain-of-Thought prompting elicits reasoning in large language models*. 28. Jan. 2022. URL: https://arxiv.org/abs/2201.11903 (besucht am 12. 10. 2024).

[22] Yifan Zhang, Yang Yuan und Andrew Chi-Chih Yao. *Meta Prompting for AI Systems*. 20. Nov. 2023. URL: https://arxiv.org/abs/2311.11482 (besucht am 12. 10. 2024).

[23] Jieyi Long. *Large language model guided Tree-of-Thought*. 15. Mai 2023. URL: https://arxiv.org/abs/2305.08291 (besucht am 14. 10. 2024).

[24] Shunyu Yao u. a. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 17. Mai 2023. URL: https://arxiv.org/abs/2305.10601 (besucht am 14. 10. 2024).

[25] Siegfried Handschuh. „Grosse Sprachmodelle“. In: Travaux du/Arbeiten aus dem Master of Advanced Studies in Archival Band 8 Nr. 1. Gesellschaft für Informatik e.V., 6. Mai 2024. URL: https://bop.unibe.ch/iw/article/view/11053/13941 (besucht am 28. 09. 2024).

[26] Sandro Hartenstein und Andreas Schmietendorf. „KI-gestützte Modernisierung von Altanwendungen: Anwendungsfelder von LLMs im Software Reengineering“. In: Softwaretechnik-Trends Band 44, Heft 2. Gesellschaft für Informatik e.V., 2024. URL: https://dl.gi.de/handle/20.500.12116/44181 (besucht am 15. 08. 2024).

Glossar

BDI-Architektur Die BDI-Achritektur ermöglicht es Agenten Ergebnisse zu übertragen und stattet die Agenten mit Weltwissen (beliefs), Ziele (desires) und Absichten (intertions). 34

Bias In der menschlichen Wahrnehmung bezeichnet Bias, eine kognitive Verzerrung oder Voreingenommenheit, die beispielsweise durch ein Vorurteil zustande kommt. In einem Neuron wird der Output ebenfalls durch dessen Wert verzerrt. Künstliche Neuronen sind mit dem Bias flexibler und erlaubt eine Verschiebung der Aktivierungsfunktion und es ist ein Output möglich, auch wenn keine Inputsignale ankommen. 13

Gradientenabstiegsverfahren Das Gradientenabstiegsverfahren (eng. Gated Recurrent Unit) ist ein 2014 eingeführtes Verfahren, um Optimierungsprobleme zu lösen. Von einem Startpunkt aus wird sich in Richtung des steilsten Abstieges bewegt. Ist ein Minimum erreicht, welcher mit einem Näherungswert übereinstimmt, ist das Verfahren abgeschlossen. 24

kollaborativ Kollaborativ lässt sich auf das lateinische Wort *collaborare* für zusammenarbeiten zurückzuführen. In einem kooperativen Multi-Agenten-System arbeiten die Agenten zusammen, um ein gemeinsames Ziel zu erreichen. Der Erfolg des Systems hängt davon ab, wie gut die Agenten zusammenarbeiten und Ressourcen teilen. 35

kompetitiv In einem kompetitiven Multi-Agenten-System hingegen arbeiten die Agenten gegeneinander, und ihre Handlungen sind oft darauf ausgelegt, ihre eigenen Vorteile auf Kosten anderer Agenten zu maximieren. 35

Overfitting Overfitting ist 22

Anhang