Diplomarbeit

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Informatik

Spoken Command Engine

Eingereicht von:	**Tobias Rechberger, 5BHIF**
	Immanuel Huber, 5BHIF
Datum:	**April 4, 2018**
Betreuer:	**Prof. Mag. Dr. Thomas Stuetz**
Projektpartner:	**flexSolution GmbH**

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.
This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 4, 2018

Tobias Rechberger, Immanuel Huber

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.
Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 4. April 2018

Tobias Rechberger, Immanuel Huber

Zusammenfassung

Die vorliegende Diplomarbeit behandelt das Interpretieren von Sprachbefehlen im Kontext eines Smart Home Systems. Zum Einen war das Ziel, Befehle zur Licht- und Temperatursteuerung richtig zu Kategorisieren und zu Parsen. Dies wurde mithilfe von Entscheidungsbäumen unter Verwendung der Softwarebibliothek Weka umgesetzt. Darüber hinaus wurde ein System geschaffen, welches sich anhand des Benutzerverhaltens automatisch an die Präferenzen des Bewohners adaptiert.

Außerdem wird gezeigt, wie durch Analyse der Stimmmerkmale - unter Zuhilfenahme eines mit Keras erstellten neuronalen Netzes - der Sprachbefehl seinem Urheber/seiner Urheberin eindeutig zugewiesen werden kann und somit der Sprecher nur anhand seines Sprachmusters identifiziert wird. Dabei wird auch darauf eingegangen, wie aus der Problemstellung eine Reinforcement Learning Umgebung erstellt wird, um das neuronale Netz effizienter zu trainieren.

Das Resultat ist eine Smart Home Schnittstelle, welche:

1. gesprochene Befehle interpretiert
2. selbstständig Benutzerpräferenzen erlernt
3. die Präferenzen mittels Identifizierung des Sprechers individualisiert

Abstract

The scope of this diploma project covers the interpretation of spoken commands in the context of a smart home system. The primary goal was to categorize and parse commands directed at the light and heating controls, which was accomplished by utilizing decision trees with the help of the software library Weka. Furthermore, a system was created which - by analyzing user behavior - automatically adapts itself to the residents' preferences.

In addition, it is shown that by analyzing the vocal features of speech it is possible to learn and identify which speaker is the originator of a command - and how this was implemented using a neural network created in Keras. It is laid out how a reinforcement learning environment was constructed and showed itself to be very fitting in the task of training the neural network more efficiently. Voice-only speaker identification plays a key role in personalizing user preferences.

The result is a smart home interface which:

1. interprets spoken commands
2. continuously learns user preferences
3. individualizes these preferences by identifying the speaker

Danksagung

An dieser Stelle möchten wir all jenen danken, die uns im Rahmen dieser Diplomarbeit begleitet haben.

Ganz besonders möchten wir Herrn Prof. Mag. Dr. Thomas Stütz danken der uns in seiner Funktion als Diplomarbeitsbetreuer, zu jederzeit bei Fragen und Problemen unterstützt hat.

Darüber hinaus möchten wir uns bei Herrn Alfred Pimminger und Herrn Christoph Pimminger bedanken, die uns mit dieser Diplomarbeit beauftragten und uns stets bei Fragen oder Problemen behilflich waren.

Danken möchten wir außerdem unseren Eltern, die uns durch ihre Unterstützung unsere Schulausbildung ermöglichten.

Inhaltsverzeichnis

1 Theorie **4**

1.1 Smart Home . . . 5
1.1.1 Was ist Smart Home? . . . 5
1.1.2 Wo ist Smart Home sinnvoll? . . . 6
1.1.3 Anwendungsbeispiel: Intelligente Heizungssteuerung . . . 6
1.1.4 Schlüsselkomponenten . . . 7
1.1.5 Sensoren . . . 7
1.1.6 Aktoren . . . 7
1.2 Maschinelles Lernen . . . 8
1.2.1 Unsupervised Learning . . . 8
1.2.2 Supervised Learning . . . 9
1.2.3 Reinforcement Learning . . . 14
1.3 Weka 3: Data Mining Software . . . 17
1.3.1 Entscheidungsbäume in Weka . . . 17
1.3.2 Alternativen zu Weka . . . 19
1.3.3 Attribute-Relation File Format - ARFF . . . 20
1.4 openHAB2 . . . 24
1.4.1 openHAB2 für Privatanwender . . . 24

2 Praxisteil **26**

2.1 Istzustand . . . 27
2.1.1 Ausgangssituation . . . 27
2.1.2 Auftrag der flexSolution GmbH . . . 27
2.2 Sollzustand . . . 28
2.2.1 Funktionale Anforderungen . . . 28
2.2.2 Nicht funktionale Anforderungen . . . 29
2.2.3 Prozesskette . . . 30
2.2.4 Prozesskette Decision Maker und Value Parser . . . 34
2.2.5 Prozesskette Keras . . . 37
2.2.6 Aktivitätsdiagramm . . . 39
2.2.7 Deployment Diagramm . . . 39
2.2.8 Klassen Diagramm . . . 40
2.3 Implementierung von Weka in Java . . . 43

2.3.1 Installation . . . 43
2.3.2 Implementierung . . . 43
2.4 Value Parser . . . 49
2.4.1 Der Prozess im Überblick . . . 49
2.4.2 Verwendete Technologien . . . 52
2.5 Decision Maker . . . 53
2.5.1 Vielseitigkeit in der Anwendung . . . 53
2.5.2 Allgemeiner Aufbau . . . 54
2.6 Speaker Identification . . . 58
2.6.1 TensorFlow . . . 58
2.6.2 Keras . . . 60
2.7 Jython . . . 70
2.7.1 Allgemein . . . 70
2.7.2 Installation . . . 70
2.7.3 Verwendung . . . 70
2.8 Web Client . . . 74

3 Anhang **77**
3.1 Protokolle . . . 77

Kapitel 1

Theorie

1.1 Smart Home

1.1.1 Was ist Smart Home?

Haushalte in denen Haushaltsgeräte, Multimediageräte und andere elektronische Komponenten miteinander kommunizieren, bezeichnet man als Smart Home. Das Smart Home zeichnet sich dadurch aus, dass Vorgänge automatisiert werden, Geräte über eine Fernsteuerung kontrolliert werden und das sich das Haus auf deine persönlichen Bedürfnisse konfiguriert. Geräte wie Heizung, Licht und Lautsprecher sind per Computer oder Smartphone komfortabel steuerbar. Immer beliebter wird die Steuerung mit Sprachbefehlen und Handzeichen, auch genannt Gesten. Das Smart Home hatte seine Anfänge schon in den 70er Jahren. Damals gab es schon kabelgebundene Varianten von vernetzen Haushalten, die über eine zentrale Steuerung überwacht werden konnten. Attraktiv wurde das Smart Home jedoch erst als zwei neue Technologien den Markt überrannten. Vergleiche [noat]

1. Bidirektionale Funkstandards, wie Bluetooth und WLAN, die es Multimediageräten und Haushaltsgeräten ermöglichen, miteinander zu kommunizieren. Sie können jetzt nicht nur Informationen empfangen, sondern auch versenden.

2. Als Smartphones und Tablets immer beliebter wurden und sich rasant verbesserten, wurden sie zum Standard für `Fernbedienungen"` im Bereich Smart Home. Der Aufwand für die Realisierung wurde gesenkt und die Heimautomatisierung wurde leichter zugänglich für den Durchschnittsbürger. Hier entstand auch der Begriff Smart Home, da zuhause viele smarte Gadgets und Geräte miteinander verbunden wurden.

Abbildung 1.1: Schwerpunkt liegt in der Interpretation des Kommandos

1.1.2 Wo ist Smart Home sinnvoll?

Mithilfe von Smart Homes wird der tägliche Tagesablauf erleichtert, indem den Bewohnern Steuertätigkeiten und Überwachungstätigkeiten abgenommen werden. Zusätzlich unterstützt uns das klug vernetzte Zuhause beim Strom sparen und schont somit die Umwelt und die eigene Geldbörse. Es gibt viele Produkttests, die zeigen, dass es zahlreiche Gadgets und Geräte zu kaufen gibt, die sich in der Praxis bewähren. Um einen kurzen Einblick zu gewähren, wird im folgenden Text ein solches Produkt genauer erklärt und dessen Effizienz gezeigt. Vergleiche [noat]

1.1.3 Anwendungsbeispiel: Intelligente Heizungssteuerung

Die Heizung wird im Sommer kaum benötigt und im Winter läuft sie ständig. Doch Frühling und Herbst sind in Mitteleuropa von Temperaturschwankungen geprägt. Um die gewünschte Wohlfühltemperatur zu erreichen, muss meist täglich am Thermostat Präzisionsarbeit durchgeführt werden. Hier sollen Smarte Wand- und Heizkörperthermostate aushelfen. Diese gibt es bereits für alle neueren Heizkörpermodelle. Man kann die smarten Thermostate ganz einfach unterwegs per Smartphone steuern, oder man verwendet ein Modell, das das Heizverhalten einfach automatisch regelt. Der Münchner Herstel-

ler `tado"` hat ein Thermostat entwickelt, das die Heizung automatisch ausschaltet, wenn niemand sich im Haus aufhält. Nähert sich ein Bewohner dem Gebäude, wird die Heizung wieder eingeschaltet. Andere Produkte setzen auf die Gewohnheiten von den Nutzern. In sogenannten Selbstlernphasen passen sie sich auf das Verhalten des Bewohners an (wann wird um wie viel die Temperatur geregelt). Nach Herstellerangaben liegt die Heizkostenersparnis bei etwa 30 Prozent. [noat]

1.1.4 Schlüsselkomponenten

Die zentrale des Smart Homes ist die sogenannte Bridge. Sie stellt den Kern des Systems dar. Alle smarten Geräte sind mit der Bridge verbunden und können somit miteinander kommunizieren. Bei den Geräten unterscheidet man zwischen Sensoren und Aktoren. Das Smart Home wird erst zum Smarten Zuhause, wenn Sensoren und Aktoren mit ihm verbunden sind und gemeinsam das Haus steuern.
Beispiel: Man stellt die Solltemperatur einer Heizung auf die gewünschte Temperatur. Ein Temperatursensor nimmt die Isttemperatur auf. Ist diese niedriger als die Solltemperatur setzt sich ein Aktor mechanisch in Bewegung, so dass Heizwasser einfließt, bis die Isttemperatur gleich der Solltemperatur ist.
Viele Geräte sind einerseits Sensoren und anderseits Aktoren. Smarte Funkstecker können beispielsweise elektronische Impulse weiterleiten und Haushaltsgeräte einschalten. Andererseits können sie den Stromverbrauch oder zum Beispiel die Raumtemperatur messen und die Werte an die Zentrale übermitteln. [noat]

1.1.5 Sensoren

Ein Sensor ist das Gegenstück zum Aktor. Er wird benötigt um physikalische Größen und chemische Effekte aufzunehmen und in ein elektronisches Signal umzuwandeln. Relevante physikalische Größen sind zum Beispiel: Gewicht, Temperatur, Lichtstärke, Druck, etc.. Sensoren kann man auch Übersetzer nennen. Sie nehmen eine physikalische Eingangsgröße auf und skalieren sie auf ein interpretierbares Ergebnis. Dazu werden sogenannte Wandler verwendet, die das Herzstück des Sensors bilden. Beispielsweise beim Geigerzähler wird eine radioaktive Strahlung analysiert und der standardisierte Wert ausgegeben. Vergleiche [noas]

1.1.6 Aktoren

Aktoren, auch genannt Aktuatoren sind das Gegenstück zu Sensoren. Sie werden benötigt um ein elektrisches Signal oder einen elektrischen Impuls in eine physikalische Größe umzuwandeln. In der Elektrotechnik werden Aktoren in Geräten wie Lampen, Lautsprechern oder Motoren eingebaut. Im Smart Home werden sie verwendet um beispielsweise Impulse des Smartphones in eine mechanische Bewegung umzusetzen. Vergleiche [noag]

1.2 Maschinelles Lernen

Carnegie Mellon University (CMU) Professor und Pionier vieler KI-bezogenen Forschungsgebiete Herbert A. Simon definierte das Lernen als „jede Änderung an einem System, die es diesem erlaubt, bei Wiederholung derselben Aufgabe oder einer anderen Aufgabe aus der gleichen Population das zweite Mal eine bessere Leistung zu zeigen“. [NM]

Maschinelles Lernen ist die Kunst, einer Rechenmaschine diese Fähigkeit des Lernens beizubringen. Im Wesentlichen gibt es drei verschiedene Formen des ML, die jeweils in unterschiedlichen Problemstellungen Anwendung finden. In den folgenden Abschnitten wird genauer auf diese Ansätze eingegangen:

- Unsupervised Learning
- Supervised Learning
- Reinforcement Learning

1.2.1 Unsupervised Learning

„Ziel des unsupervised Learning Ansatz ist es, aus den Daten unbekannte Muster zu erkennen und Regeln aus diesen abzuleiten.“ [Gor] Dabei werden die Daten anhand der erschlossenen Regeln in Cluster zusammengefasst, um daraus Informationen zu beziehen. Im Gegensatz zum supervised Learning sind die Daten nicht mit Labels behaftet. Da im Rahmen der Diplomarbeit unsupervised Learning keine Verwendung findet, wird nicht weiter auf diese Vorgehensweise eingegangen. In Abbildung 1.2 ist das Clustern anhand eines allgemeinen Beispiels grafisch dargestellt.

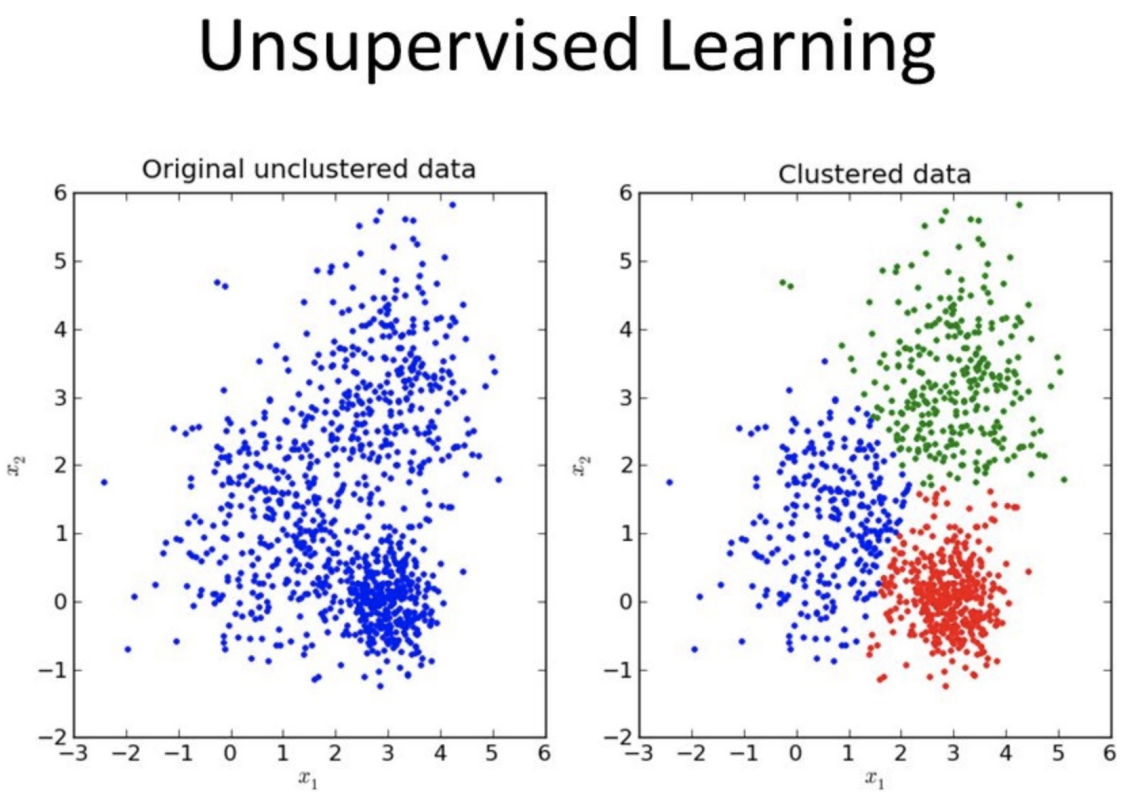


Abbildung 1.2: Unsupervised Learning - das Clustern von Daten [fra]

1.2.2 Supervised Learning

„Beim supervised Learning geht es darum eine Funktion zu finden, mit der ungesehene bzw. unbekannte Beobachtungen eines Datensets einer Klasse oder einem Wert zugewiesen werden können. Hierfür werden die Daten mit einem sogenannten Label versehen.“ [Gor] Die häufigsten Anwendungsgebiete des supervised Learning Ansatzes sind die Klassifizierung und die Regression.

Anwendungsgebiete

Klassifizierung „Das Ziel bei der Klassifikation ist es, eine Klassenbezeichnung vorherzusagen, ergo eine Auswahl aus einer vorgegebenen Liste von Möglichkeiten zu treffen. Klassifikationsverfahren unterteilt man in binäre Klassifikation (z.B. JA/NEIN o. Ist diese E-Mail Spam?) und Klassifikation mehrerer Klassen (z.B. Erkennung und Zuordnung der unterschiedlichen Personen in meiner Bilddatenbank).“ [Dur]

Regression „Bei Regressionsproblemen ist das Ziel, eine kontinuierliche Größe oder Fließkommazahl im Programmierjargon vorherzusagen. Das jährliche Einkommen einer

Person aus Bildungsgrad, Alter und Wohnort vorherzusagen, ist ein Beispiel für ein Regressionsproblem.“ [Dur]

Abbildung 1.3 veranschaulicht den Unterschied zwischen einer Klassifizierung und einer Regression.

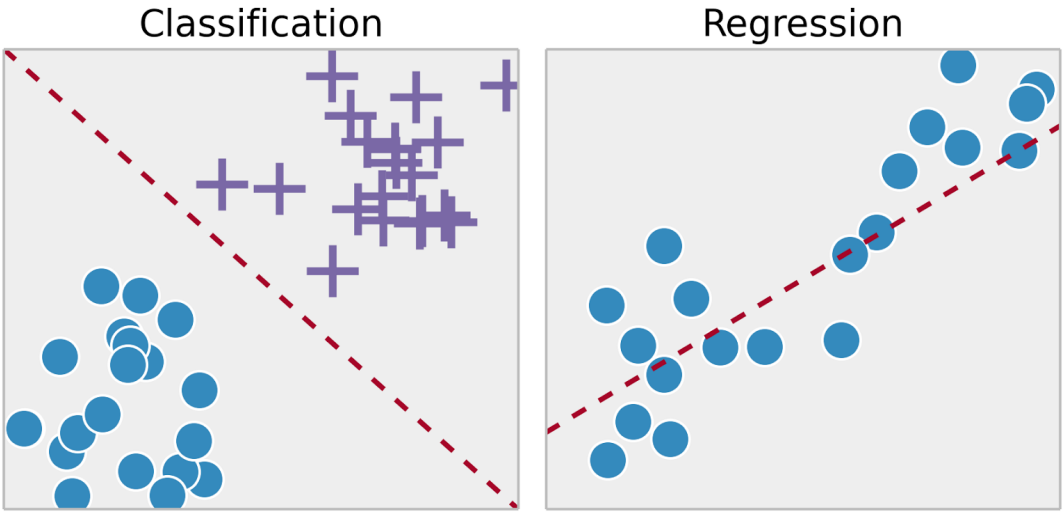


Abbildung 1.3: Klassifizierung und Regression von Beispieldaten [Kor]

Künstliche neuronale Netze

Künstliche neuronale Netze stellen die Lösung auf eine Vielzahl an Klassifizierungsproblemen dar. Es wird dabei das menschliche neuronale Netz abstrahiert und nachgeahmt. „Neuronale Netze bestehen aus mehreren Neuronen. Diese Neuronen werden auch als Units, Einheiten oder Knoten bezeichnet. Sie dienen dazu, Informationen aus der Umwelt oder von anderen Neuronen aufzunehmen und an andere Units oder die Umwelt in modifizierter Form weiterzuleiten.“ [Neud] In Abbildung 1.4 wird der allgemeine Aufbau eines neuronalen Netzes anhand eines Beispiels illustriert.

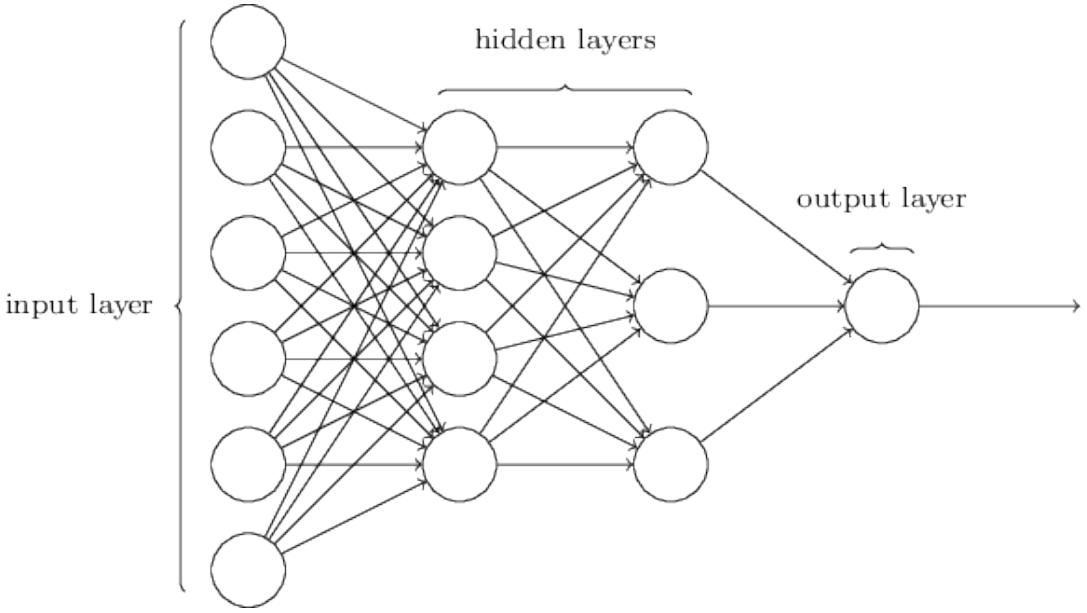


Abbildung 1.4: Allgemeiner Aufbau eines neuronalen Netzes [Nie]

In jedem neuronalen Netz gibt es einen Input Layer (dt. Eingabeschicht). Hier werden die Eingabedaten in das Netz eingespeist. Im Beispiel eines Netzes zur Erkennung von Ziffern in einem Bild wären das die einzelnen Pixel der Grafik. Am anderen Ende des Netzes befindet sich der Output Layer (dt. Ausgabeschicht). Um zum vorhergehenden Exempel zurückzukehren wäre das die ermittelte Ziffer. Diese beiden essenziellen Schichten verbindet eine beliebige Anzahl an sogenannten Hidden Layers (dt. versteckte Schichten).

„Units sind miteinander durch Kanten verbunden. Die Stärke der Verbindung zwischen zwei Neuronen wird durch ein Gewicht ausgedrückt. Je größer der Absolutbetrag des Gewichtes ist, desto größer ist der Einfluss einer Unit auf eine andere Unit. [...] Lernen wird bei neuronalen Netzen zumeist als Gewichtsveränderungen zwischen den Einheiten definiert. Wie die Gewichtsveränderung genau erfolgt ist abhängig von der verwendeten Lernregel.“ [Neue]

Künstliche Neuronen

In Abbildung 1.5 ist der Aufbau eines künstlichen Neurons dargestellt.

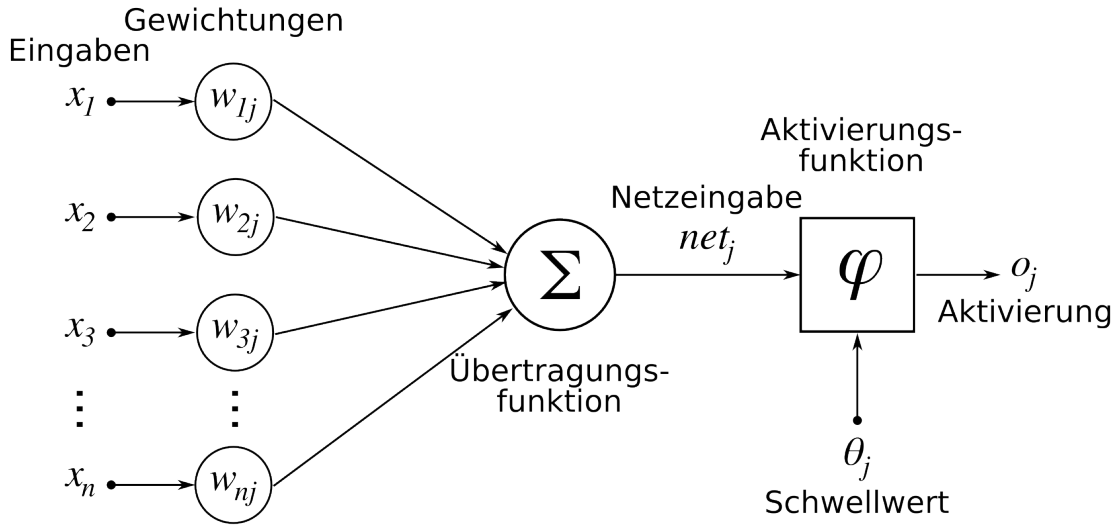


Abbildung 1.5: Schema eines künstlichen Neurons [Knn]

Ein Neuron hat n Eingabeparameter, die jeweils mit der Gewichtung w_{ij} versehen sind. Diese Werte werden mit folgender Übertragungsfunktion zu einem Wert, der Netzeingabe net_j, zusammengefasst:

$$net_j := \sum_{i=1}^{n} x_i * w_{ij} \tag{1.1}$$

Die Netzeingabe net_j wird anschließend der Aktivierungsfunktion ϕ übergeben, welche dann bei Überschreitung des Schwellwertes θ_j den Output der Neurone, die Aktivierung o_j, liefert.

Aktivierungsfunktionen

„Die Aktivitätsfunktion (Transferfunktion, Aktivierungsfunktion) stellt den Zusammenhang zwischen dem Netzinput und dem Aktivitätslevel eines Neurons dar.“ [Neua] In Abbildung 1.6 sind drei Beispiele von häufig verwendeten Aktivierungsfunktionen grafisch dargestellt:

- Sigmoid: wird angewendet, um die Netzeingabe in einen Wert zwischen 0 und 1 zu konvertieren
- Tanh: wird angewendet, um die Netzeingabe in einen Wert zwischen -1 und 1 zu konvertieren
- Rectified Linear Unit (ReLu): wird angewendet, um nur positive Werte zu übernehmen

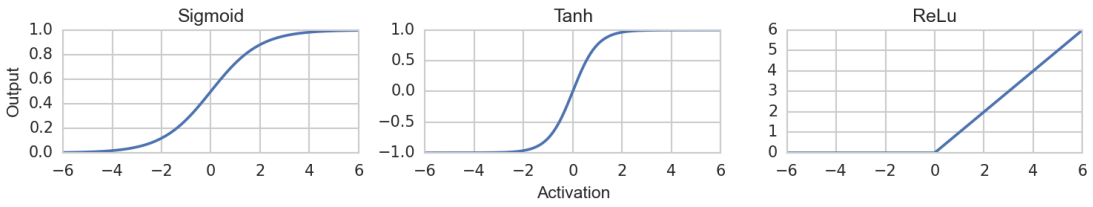


Abbildung 1.6: Häufig verwendete Aktivierungsfunktionen [Bal]

Die Lernregel

„Um die Gewichte in der Trainingsphase zu modifizieren, benötigt man eine Lernregel, die angibt, wie die Veränderungen vorgenommen werden sollen. Eine Lernregel stellt dabei einen Algorithmus dar, der darüber Auskunft gibt, welche Gewichte des neuronalen Netzes wie stark erhöht oder reduziert werden sollen.“ [Neub] Unter Anderem werden folgende Lernregeln häufig angewendet:

- Stochastic Gradient Descent (SDG)
- Adam
- AdaGrad
- Momentum
- Nesterov Momentum

Da bei unterschiedlichen Datenstrukturen und Anwendungsfällen die Lernregeln unterschiedlich abschneiden, kann keine allgemein „beste“ Lernregel ausgemacht werden. Häufig ergibt sich das Ausprobieren und Vergleichen diverser Lernregel als ausreichende Strategie, die beste Lernregel für die jeweilige Situation zu finden.

Rekurrente Netze

„Rekurrente Netze sind dadurch gekennzeichnet, dass Rückkopplungen von Neuronen einer Schicht zu anderen Neuronen derselben oder einer vorangegangenen Schicht existieren. Damit sollen zumeist zeitlich codierte Informationen in den Daten entdeckt werden.“ [Neuc] Häufig werden in rekurrenten neuronalen Netzen sogenannte Long Short-Term Memory (LSTM) Neuronen angewendet. LSTM Zellen können sich Informationen über einen längeren Zeitraum „merken“ und somit Zusammenhänge über einem breiteren Kontext erkennen.

Im oben genannten Beispiel eines Ziffernparsers ist ein reguläres neuronales Netz vollkommen ausreichend, da die erkannte Zahl nicht von der zeitlichen Reihenfolge mehrerer interpretierter Bilder abhängt. Sollte jedoch der Inhalt zu einem natürlichen Text ermittelt werden, so ist die Reihenfolge der Wörter von wesentlicher Bedeutung. Für solche Anwendungsfälle eignet sich ein rekurrentes neuronales Netz. Abbildung 1.7 zeigt den allgemeinen Aufbau der rekurrenten Schicht eines neuronalen Netzes.

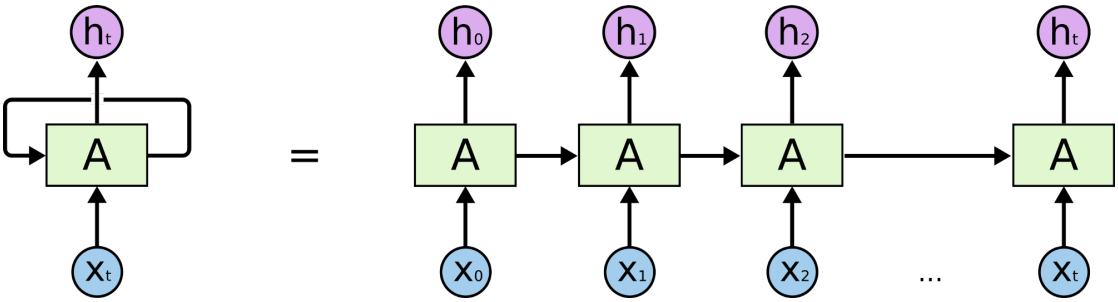


Abbildung 1.7: Schicht eines rekurrenten neuronalen Netzes [Rad]

A repräsentiert ein reguläres neuronales Netzwerk, x_i den Input-Vektor und h_i den Output-Vektor zum Zeitpunkt i. Es ist zu erkennen, dass das Ergebnis einer Iteration zusätzlich als Parameter in die nächste Iteration einfließt.

1.2.3 Reinforcement Learning

„Reinforcement Learning ist die Lernmethode des Kindes, das das Gehen lernt, auf Maschinen angewandt [...] und basiert hauptsächlich auf die Kommunikation zwischen lernenden “Agent” und der Umgebung. Das Konzept versucht sein (Lern-)Ziel zu erreichen, indem es den Zustand der Umgebung nach einer gewählten Aktion, in Hinblick auf das Ziel, belohnt und durch die gesammelten Erfahrungen die zukünftigen Aktionen so wählt, dass die Belohnung zu maximiert wird.“ [Her] Abbildung 1.8 veranschaulicht den Zyklus eines Reinforcement Learning Systems.

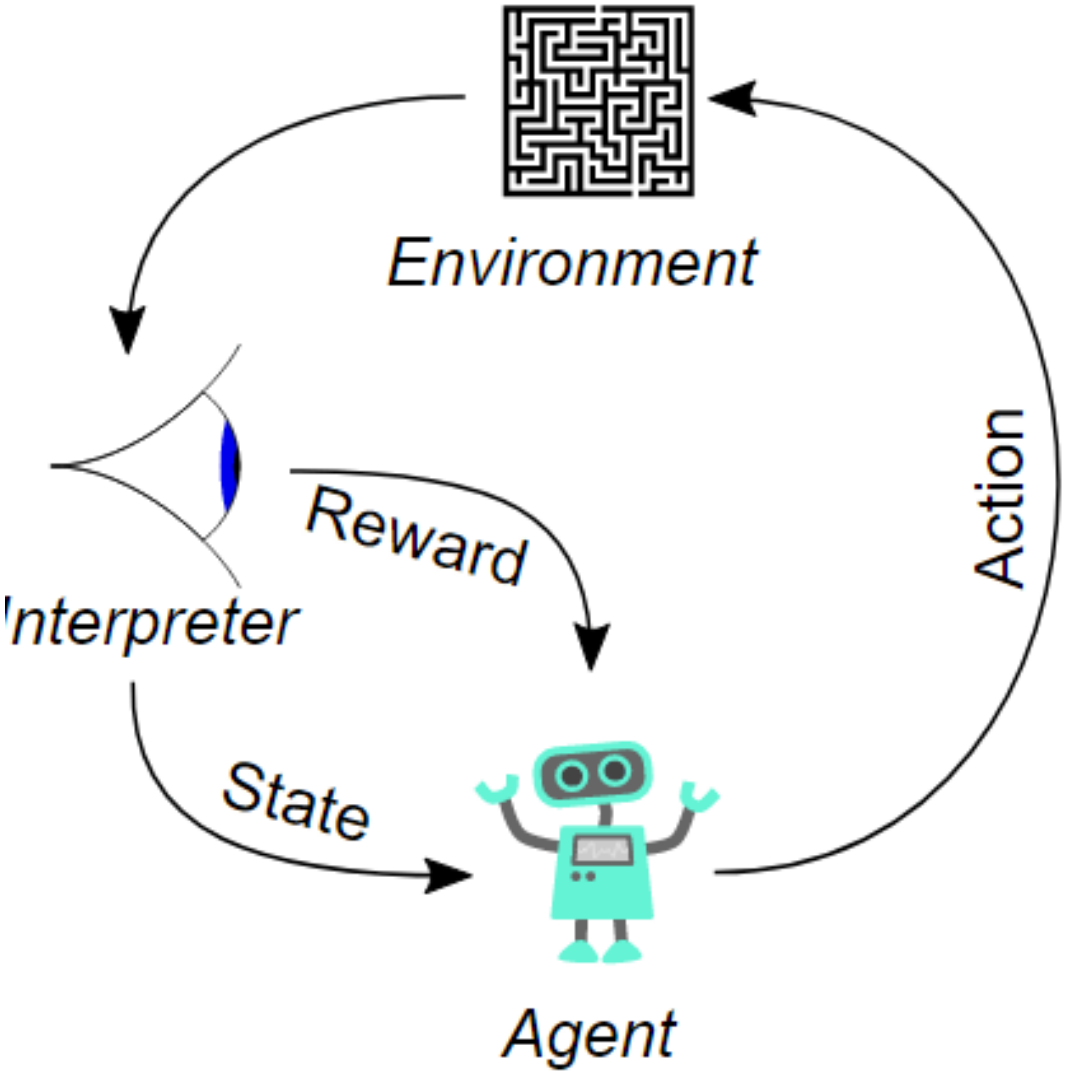


Abbildung 1.8: Zyklus eines Reinforcement Learning Systems [Gui]

Folgende Schritte werden pro Iteration durchlaufen:

1. Der Agent observiert den aktuellen Zustand (State) der Umgebung (Environment)
2. Der Agent berechnet sich die bestmögliche Aktion für den gegebenen Zustand
3. Der Agent führt die ausgewählte Aktion durch
4. Der Interpreter überprüft die Auswirkung der vom Agenten vollzogenen Aktion

5. Der Interpreter ermittelt anhand dessen die Belohnung (Reward) des Agenten
6. Der Agent adaptiert seine Strategie (Policy) auf Basis der erhaltenen Belohnung

1.3 Weka 3: Data Mining Software

Weka 3 ist eine Sammlung von open source machine learning Algorithmen, die im Bereich data mining verwendet werden. Die Algorithmen können entweder über ein GUI, dass von Weka zur Verfügung gestellt wird, aufgerufen werden oder direkt über den eigenen Java code. Unter anderem enthält Weka Hilfsmittel für Datenvorbereitung, Datenklassifizierung, Regression und Datengruppierung. Es besteht auch die Möglichkeit Weka im Bereich big data zu verwenden. Weka wurde an der Universität Waikato in Neuseeland entwickelt.
Die Ziele von Weka sind:

- maschinelles Lernen für die Öffentlichkeit zugänglich zu machen
- neue Algorithmen im Umfeld machine learning zu entwickeln
- für Fortschritt in diesem Bereich zu sorgen und einen Anteil dazu beizutragen

1.3.1 Entscheidungsbäume in Weka

Damit ein lernender Algorithmus in einer Applikation, die von vielen Benutzern verwendet wird, brauchbar ist, muss er ein relevantes Ergebnis zurückliefern, auch wenn Test Daten nicht vollständig sind oder zufällige Fehler enthalten. Bei decision trees ist dies meistens gegeben und sie sind dennoch leicht zu verstehen und anzuwenden.
Um einen Entscheidungsbaum zu erstellen wird zuerst aus den Trainingsdaten ein Modell generiert, dass in Form eines Baumes abgebildet wird. Das Ziel ist mit so wenig Entscheidungen, wie möglich, die beste mögliche Klassifizierung von Testdaten zu erreichen. Vergleiche [noaf]
Beispiel: Es wird versucht, anhand der Attribute CountryOfOrigin, BigStar und Genre den Erfolg eines Filmes vorherzusagen. Abbildung 1.9

Aus diesen Trainingsdaten muss nun ein predictive model anhand eines decision tree Algorithmus generiert werden. Diesen Baum könnte man grafisch wie folgt darstellen:
Abbildung 1.10 Jede Ebene teilt die Daten in verschiedene Attribute. Die Knoten die keine Blätter sind bilden die Attribute ab. Die Blätter zeigen dagegen den Wert der vorhergesehen wird.
Einfache decision tree Algorithmen erfolgen immer in einem ähnlichen Ablauf. Als erstes wird mit den Trainingsdaten begonnen. Danach muss das beste Attribut bzw. der beste Wert ausgewählt werden. Dieser Wert wird anhand des `"best split"` Verfahrens, das im nächsten Kapitel genauer erläutert wird, ermittelt. Dieser Vorgang wird rekursiv auf die daraus gewonnenen Ergebnisse angewandt bis der Baum zu groß wird, oder die Anzahl der Daten zu klein ist.

Best Split Verfahren

Die Schwierigkeit beim Erstellen des Baumes ist, zu entscheiden, nach welchem Attribut aufgeteilt werden muss, um den `"best split"` zu erhalten. Hierbei wird auf das

CountryOfOrigin	BigStar	Genre	Success/Failure
USA	yes	scifi	Success
USA	no	comedy	Failure
USA	yes	comedy	Success
Europe	no	comedy	Success
Europe	yes	scifi	Failure
Europe	yes	romance	Failure
Australia	yes	comedy	Failure
Brazil	no	scifi	Failure
Europe	yes	comedy	Success
USA	yes	comedy	Success

Abbildung 1.9: Trainingsdaten für den Entscheidungsbaum

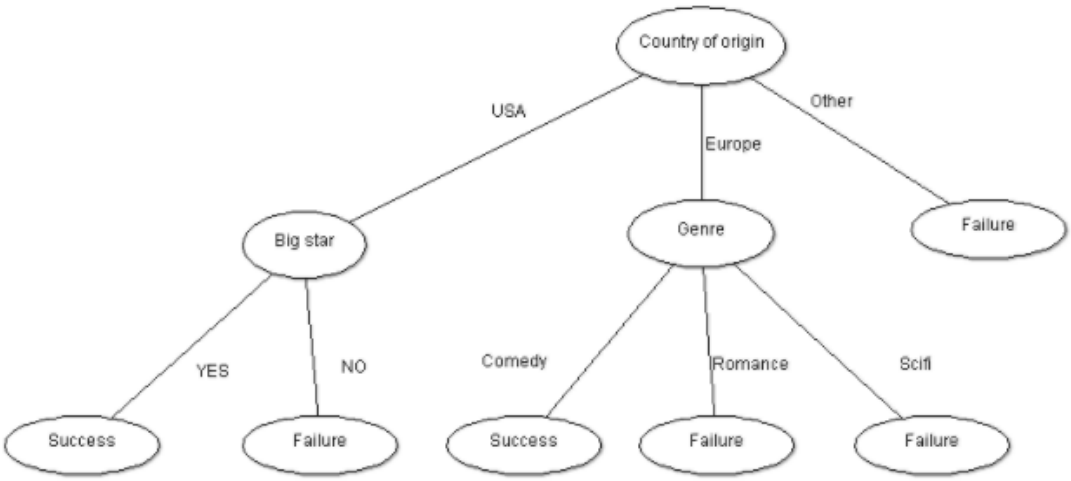


Abbildung 1.10: Trainingsdaten abgebildet in Form eines Entscheidungsbaumes

“Information Gain Concept“ aufgebaut. Der Informationsgehalt ist der Unterschied der Entropie vor und nach einem Split. Die Entropie ist die Ungewissheit in einer Information. Sie wird berechnet mit:

- Entropy = -pP * log2(pP) -pN * log2(pN)
- pP – the proportion of positive (training) examples
- pN – the proportion of negative (training) examples
- pP - Anteil der positiven Trainingsdaten
- pN - Anteil der negativen Trainingsdaten

Beim Beispiel ist die Anfangsentropie 1, da es 5 “successes“ und 5 “failures“ gibt. Der Algorithmus beginnt mit der Berechnung des Informationsgehaltes beim Attribut “Country Of Origin“. Im Falle der USA gibt es 3 successes und 1 failure. Das heißt 75% sind positiv und 25% negativ, oder 0.75 und 0.25. Nun wird die Entropie mit dem Logarithmus zur Basis 2 wie folgt berechnet. Dies wird für alle Länder bzw. Kontinente durchgeführt. Die Entropie für Country of Origin ergibt sich aus der Entropie aller Länder und Kontinente, multipliziert mit deren prozentualen Anteil an den Trainingsdaten.

- Entropy(USA) = -0.75 * log2(0.75) – 0.25 * log2(0.25) = 0.811
- Entropy(Europe) = 1.0 (2 of each)
- Entropy(Other) = 0.0 (2 Failures)
- Entropy(CountryOfOrigin) = 0.4 * 0.811 + 0.4 * 1.0 + 0.2 * 0.0 = 0.7244
- Informationsgehalt(CountryOfOrigin) = 1.0 – 0.7244 = 0.2756

Wenn man dieses Verfahren für die anderen Attribute anwendet, kommt man auf die Werte:

- Informationsgehalt für “BigStar” = 0.03
- Informationsgehalt für “Genre” = 0.16

Da der Informationsgehalt von “CountryOfOrigin“ am höchsten ist, wird dieses Attribut als erstes ausgewählt. Der Baum wird fertig generiert, indem dieses Verfahren für die restlichen Attribute durchgeführt wird.

1.3.2 Alternativen zu Weka

Rapid Miner

Rapid Miner, zuvor YALE genannt (Yet Another Learning Environment), ist seit 2001 in Entwicklung an der Universität Dortmund im Bereich künstliche Intelligenz. Rapid Miner ist eine Software für maschinelles Lernen. Die Experimente werden mit XML

beschrieben und in einer grafischen Benutzeroberfläche entwickelt. Die Software enthält Möglichkeiten um Forschungsanwendungen sowie Industrieanwendungen zu entwickeln. Das Rapid Miner Graphical User Interface ist grundsätzlich kostenlos, jedoch ist der Benutzer auf 10.000 Datensätze beschränkt und der logische Prozess ist limitiert. Alle maschinelle Algorithmen von Weka sind im Rapid Miner enthalten. Zusätzlich bietet der Rapid Miner Algorithmen im Bereich Stimmungsanalyse. Eine Java Implementierung ist wie in Weka möglich. Vergleiche [noae] [noao]

KNIME

KNIME (Konstanz Information Miner) ist eine open source Software, entwickelt von Software-Entwicklern aus dem Silicon Valley an der Universität Konstanz, die für interaktive Datenanalyse verwendet werden kann. Die machine learning und Data-Mining Software bietet, wie viele ihrer Art, eine grafische Benutzeroberfläche, mit der ohne viel "Know How"gearbeitet werden kann. KNIME wird meist in pharmazeutischer Forschung verwendet, jedoch findet sich auch Verwendung im Sektor Business Intelligence und Finanzdatenanalyse. Eine Implementierung in Java ist möglich. Vergleiche [noak]

1.3.3 Attribute-Relation File Format - ARFF

Allgemein

Ein Attribute Relation File Format kurz ARFF ist ein ASCII Textfile, das eine Liste von Instanzen mit den gleichen Attributen beschreibt. ARFF Files wurden auf der University of Waikato mit der Weka machine learning Software entwickelt. Sie wurden entworfen um die Struktur der Trainingsdaten für machine learning algorithmen zu beschreiben. Vergleiche [noac]
Man teilt ARFF Files in zwei verschiedene Teile. Der erste Teil ist die Header Information. Der zweite Teil sind die Daten. Der Header beinhaltet den Namen der Relation, alle Attribute und ihre Datentypen. Ein Header könnte beispielsweise so aussehen: Abbildung 1.11

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature number
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
```

Abbildung 1.11: Header Teil des ARFF Files weather

Die Daten des ARFF Files können wie folgt aussehen: Abbildung 1.12

```
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Abbildung 1.12: Data Teil des ARFF Files weather

ARFF Header

Der ARFF Header Teil beinhaltet die Deklarationen für die Attribute und Relation.

Relation

Die Relation ist die erste Zeile im ARFF File. Sie muss wie folgt aussehen: 1.13

```
@relation <relation-name>
```

Abbildung 1.13: Relation eines ARFF Files

Kommentare werden mit % gekennzeichnet. Bei den Feldern @RELATION, @ATTRIBUTE und @DATA muss nicht auf die Groß.- Kleinschreibung geachtet werden.

Attribut

Alle Attribute werden hintereinander in der Datei angeordnet. Jedes Attribut Feld hat sein eigenes @ATTRIBUTE und bestimmt somit eindeutig seinen Namen und den Datentyp (Abbildung 1.131.14). Je nachdem wie die Attribute angeordnet sind, wird die Position im Daten Bereich festgelegt. Ist ein Attribut an zweiter Stelle, so ist es der zweite Wert bei den Daten.

```
@attribute <attribute-name> <datatype>
```

Abbildung 1.14: Attribut eines ARFF Files

Für den Platzhalter <attribute-name> muss ein “alphabetic character“ verwendet werden. Für <datatype> gibt es vier verschiedene Möglichkeiten:

- numeric
- <nominal-specification>
- string
- date

Numerische Attribute

Numerische Attribute können reelle oder integer Zahlen sein.

Nominale Attribute

Bei nominalen Attributen werden die möglichen Werte vordefiniert: {<nominal-name1>, <nominal-name2>, <nominal-name3>}. Zum Beispiel für die Aussicht beim Wetter:

```
@attribute outlook {sunny, overcast, rainy}
```

Abbildung 1.15: Nominales Attribut outlook

String Attribute

String Attribute ermöglichen das Verwenden von zum Beispiel ganzen Sätzen. Dies ist sehr hilfreich in text-mining Anwendungen, da man mit Hilfe von Filtern (z.B StringToWordVektor) diese in mehrere Attribute aufteilen kann.

```
@attribute <name> string
```

Abbildung 1.16: Deklaration eines String Attributes

Date Attribute

Die Deklaration des Date Attribut sieht wie folgt aus:

```
@attribute <name> date [<date-format>]
```

Abbildung 1.17: Deklaration eines Date Attributes

Beim <date-format> Platzhalter kann das gewünschte Datumsformat eingetragen werden. Es werden die ISO-8601 Standards für die Formatierung verwendet.

ARFF Data

Der Data Bereich beginnt mit der Annotation @data. Danach werden alle Datensätze hintereinander aufgelistet. Jeder Datensatz befindet sich in einer eigenen Zeile. Die Attribute für jede Instanz werden mit Kommas getrennt. Die Anordnung der Attribute muss gleich sein wie die Reihenfolge der Deklaration im Header Teil. Wenn ein Attribut fehlt, wird es mit einem Fragezeichen ? gekennzeichnet.

```
@data
sunny,?,85,FALSE,no
```

Abbildung 1.18: Deklaration eines Datensatzes mit fehlendem Wert

1.4 openHAB2

openHAB2 steht für open Home Automation Bus und ist eine open source Software, die es ermöglicht Smart Home Geräte von unterschiedlichen Herstellern mit verschiedenen Protokollen von einer Zentrale aus zu steuern. Die proprietäre Software der einzelnen Hersteller wird nicht mehr benötigt, da openHAB2 fast alle Smart Home Protokolle unterstützt. Dies ermöglicht es, dass man nicht mehr auf einen Hersteller angewiesen ist und man nun die besten Produkte der verschiedensten Unternehmen kombinieren kann, ohne eine großen Aufwand bei der Zusammenführung zu haben. openHAB2 ist Plattform unabhängig und kann auch auf weniger leistungsstarker Hardware, wie einem Raspberry PI 3, betrieben werden. Vergleiche [noam]

1.4.1 openHAB2 für Privatanwender

openHAB2 ist im Gegensatz zur älteren Version openHAB weit aus einsteigerfreundlicher. Es müssen keine technischen Vorkenntnisse vorhanden sein, da das HABPanel 1.19 leicht zu bedienen ist. Die Entwickler von openHAB2 legen einen großen Wert auf die einfache Bedienbarkeit, deswegen soll diese in zukünftigen Versionen noch besser werden. Erweiterungen kann man ganz einfach hinzufügen und sie werden automatisch in das openHAB Graphical User Interface eingebunden. Die GUI ist für Tablets ausgelegt, jedoch gibt es auch eine Windows 10 Applikation. Grundsätzlich ist das GUI webbasiert und openHAB2 stellt einen Webserver zur Verfügung, der bei der Installation automatisch eingerichtet wird. Die meisten Einrichtungssituationen können mit der grafischen Oberfläche die openHAB2 zur Verfügung stellt abgedeckt werden. Treten jedoch spezial Anforderungen auf, oder will der Anwender das System selbst konfigurieren, ist dies auch möglich. Über Textkonfigurationsdateien kann das GUI individuell eingestellt werden. Diese Dateien sind sehr ähnlich wie in openHAB1 und deshalb nicht sehr schwer zu verstehen. Vergleiche [noap]

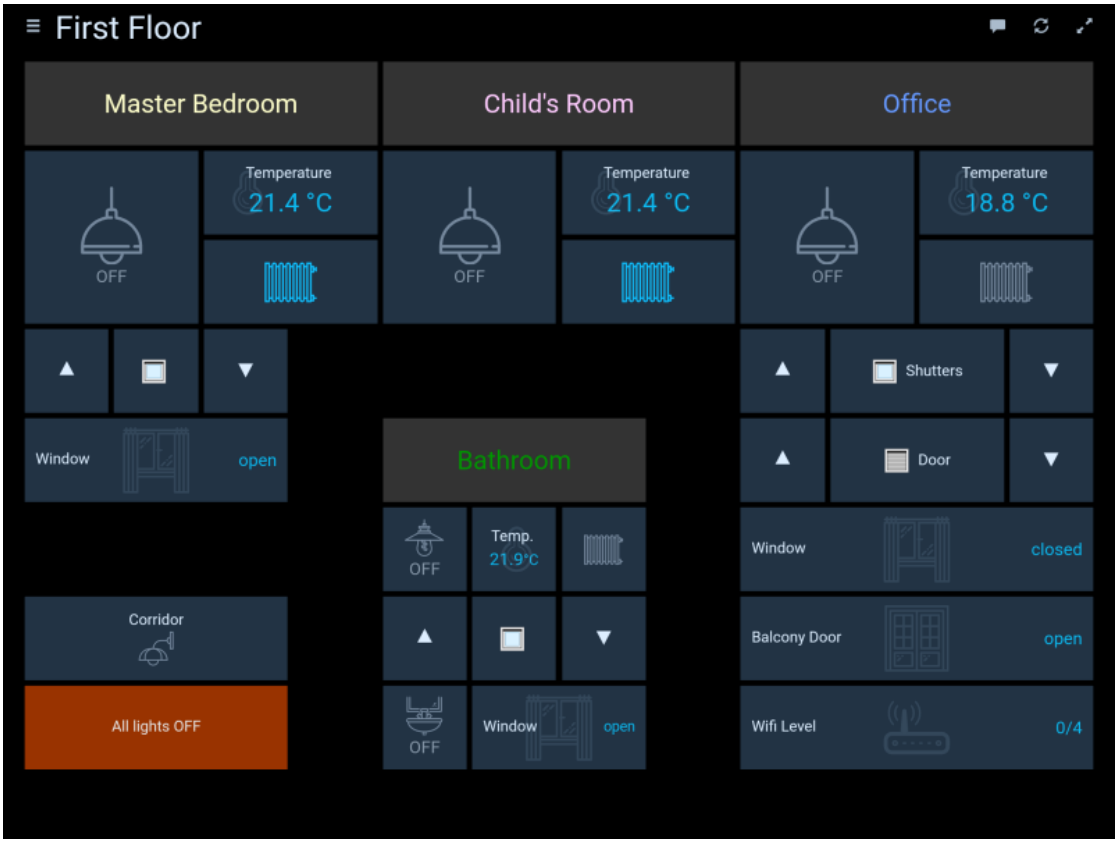


Abbildung 1.19: Schwerpunkt liegt in der Interpretation des Kommandos

Kapitel 2

Praxisteil

2.1 Istzustand

2.1.1 Ausgangssituation

Es wird von einem deutschsprachigen Text ausgegangen, der bereits von einem Spracherkennungssystem umgesetzt wurde. Im Sinne der Orthographie soll der Text semantisch untersucht werden, in der die Bedeutung bzw. die Schlüsselwörter in den Mittelpunkt gestellt werden. Auf Basis von vorgegebenen Schlüsselwörtern sollen Befehlswörter resultieren. Es liegt der Bereich SmartBuilding im Fokus und dementsprechend sind Schlüsselwörter und Befehlswörter zu wählen.
Die flexSolution GmbH produziert individual-Lösungen im Bereich Smart Homes. Hier kommt die Spoken Command Engine zum Einsatz. Von der Spoken Command Engine wird erwartet, dass es zusätzlichen Komfort für den Endnutzer des Smart Homes bietet. Alle Vorteile die das Smart Home zur Verfügung stellt wie Heizungssteuerung, Lichtmanagement, etc. werden zurzeit über eine Smartphone Applikation gesteuert. Dies ist sehr unvorteilhaft, da man das Smartphone nicht immer zur Hand hat und es mühsam ist, extra die Anwendung zu starten um das Licht einzuschalten. Oft ist es auch der Fall, dass man keine Hand frei hat, da man gerade kocht oder sein Baby füttert.

2.1.2 Auftrag der flexSolution GmbH

Die flexSolution GmbH ist ein mittelständisches Unternehmen in Neuhofen an der Krems.
Sie verkaufen Produkte im Bereich Maschinen.- und Gebäudeautomatisierung. Der Geschäftsführer Herr Pimminger Alfred leitet auch das ältere Unternehmen SMA - Sondermaschinen- u. Anlagenbau GmbH. Die flexSolution GmbH wurde am 27.05.2010 gegründet und hat vier Mitarbeiter. Die SMA GmbH wurde am 21.05.1997 gegründet und hat 21 Mitarbeiter.

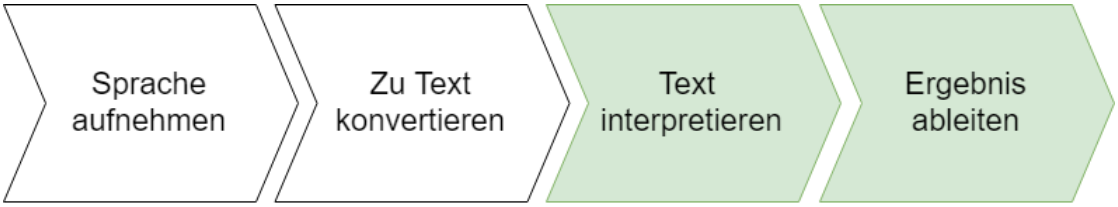


Abbildung 2.1: Schwerpunkt liegt in der Interpretation des Kommandos

2.2 Sollzustand

2.2.1 Funktionale Anforderungen

Befehl interpretieren

Das Kommando beziehungsweise der Befehl, der vom Benutzer gegeben wird, soll inhaltlich und sinnhaft verstanden werden. Hierbei sollen nicht nur vorgefertigte Kommandos erkannt werden, sondern das System soll Alltagssprache verstehen und interpretieren können. Neue Kategorien wie Radio, Fernseher, Luft sollen im späteren Verlauf leicht erweiterbar sein. Datumsangaben wie Wochentage (am Montag, am Freitag) und Uhrzeitangaben (um 17 Uhr, um 7:30) sollen interpretiert werden. Hier wird darauf Wert gelegt, dass das System auch komplexere Angaben wie zum Beispiel: übernächsten Montag um 8 Uhr erkennt.

Sprecher erkennen

Ein Smart Home wird erst zum smarten Zuhause, wenn es auf die Bedürfnisse jedes einzelnen Bewohners eingehen kann. Um dies zu ermöglichen, muss die Sprachdatei des aufgenommenen Befehls auf Stimmmerkmale überprüft werden, um den Sprecher zu erkennen. Das System soll ohne großen Aufwand einen neuen Bewohner erkennen und hinzufügen können.

Präferenzen erkennen

Alle Befehle, die über die Sprachsteuerung getätigt wurden, werden in einer Datenbank abgespeichert. Sämtliche Daten, die dem System zum Zeitpunkt der Befehlsgabe bekannt sind, wie zum Beispiel: Außentemperatur, Innentemperatur, Uhrzeit, Wochentag, etc. sollen festgehalten werden. Zusätzlich wird noch der Sprecher, der das Kommando gab, abgespeichert. Aus diesen Daten können nun für jeden individuellen Bewohner Schlüsse über wiederkehrende Ereignisse, wie zum Beispiel jeden Dienstag um 6 Uhr wird das Badezimmer vorgeheizt, gezogen werden.

Präferenzen verwenden

Für jeden Benutzer sollen zu jeder Zeit seine persönlichen Präferenzen festgestellt werden können. Das Smart Home soll automatisch seine Aktoren auf die Vorlieben des Bewohners einstellen. Alle Präferenzen für jede Kategorie müssen ständig abgerufen werden können. Werden Befehle ohne Skalar gegeben wie zum Beispiel: Mir ist zu kalt, wird automatisch der Sprecher erkannt und die Temperatur wird an seine Gewohnheit angepasst.

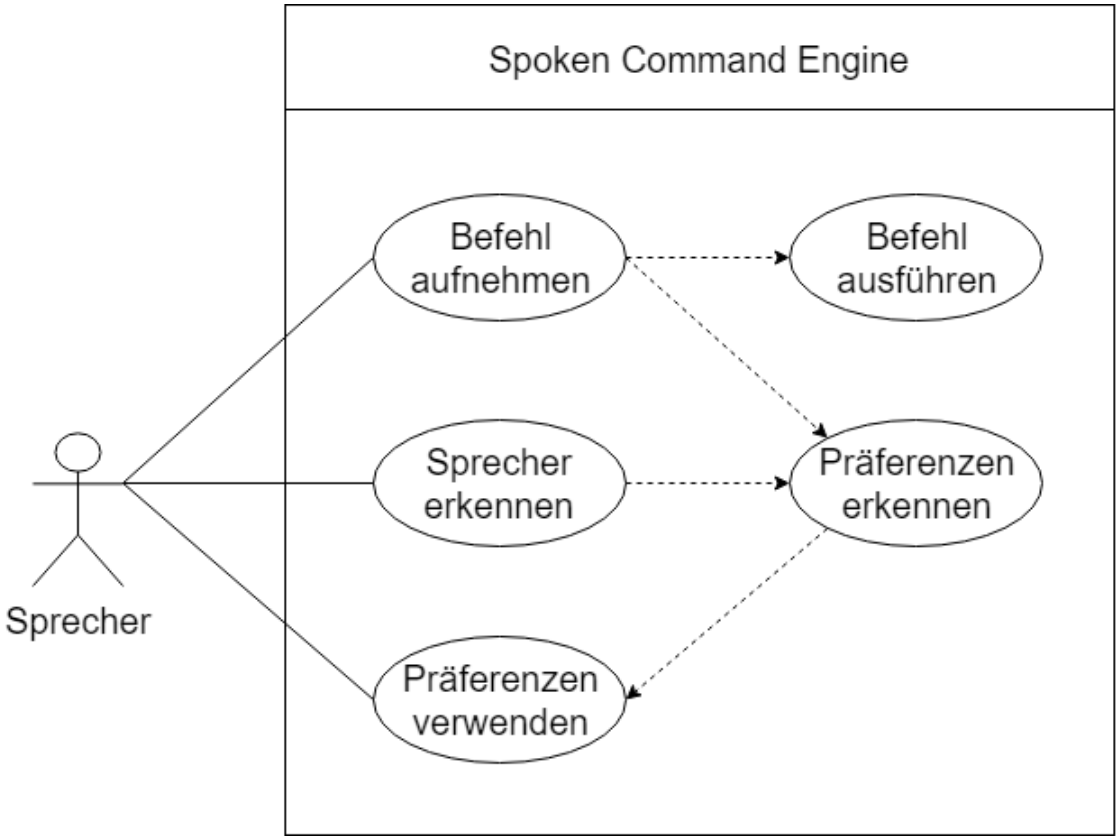


Abbildung 2.2: Use Case Diagramm des Systems

2.2.2 Nicht funktionale Anforderungen

Latenz

Ein Benutzer eines Systems empfindet einen sogenannten Ïnput Lagäb einer Verzögerung von etwa 150 Millisekunden. Wenn man ein Licht einschaltet und es leuchtet erst 150 Millisekunden nach dem betätigen der Taste, kann man von einem Input Lag sprechen. Dies ist sehr unangenehm für den Benutzer und erzeugt ein unwohles Gefühl. Deshalb ist es wichtig, dass das System eine sofortige Rückmeldung an den Benutzer gibt. Somit ist ein angenehmeres Bedienen gewährleistet.

2.2.3 Prozesskette

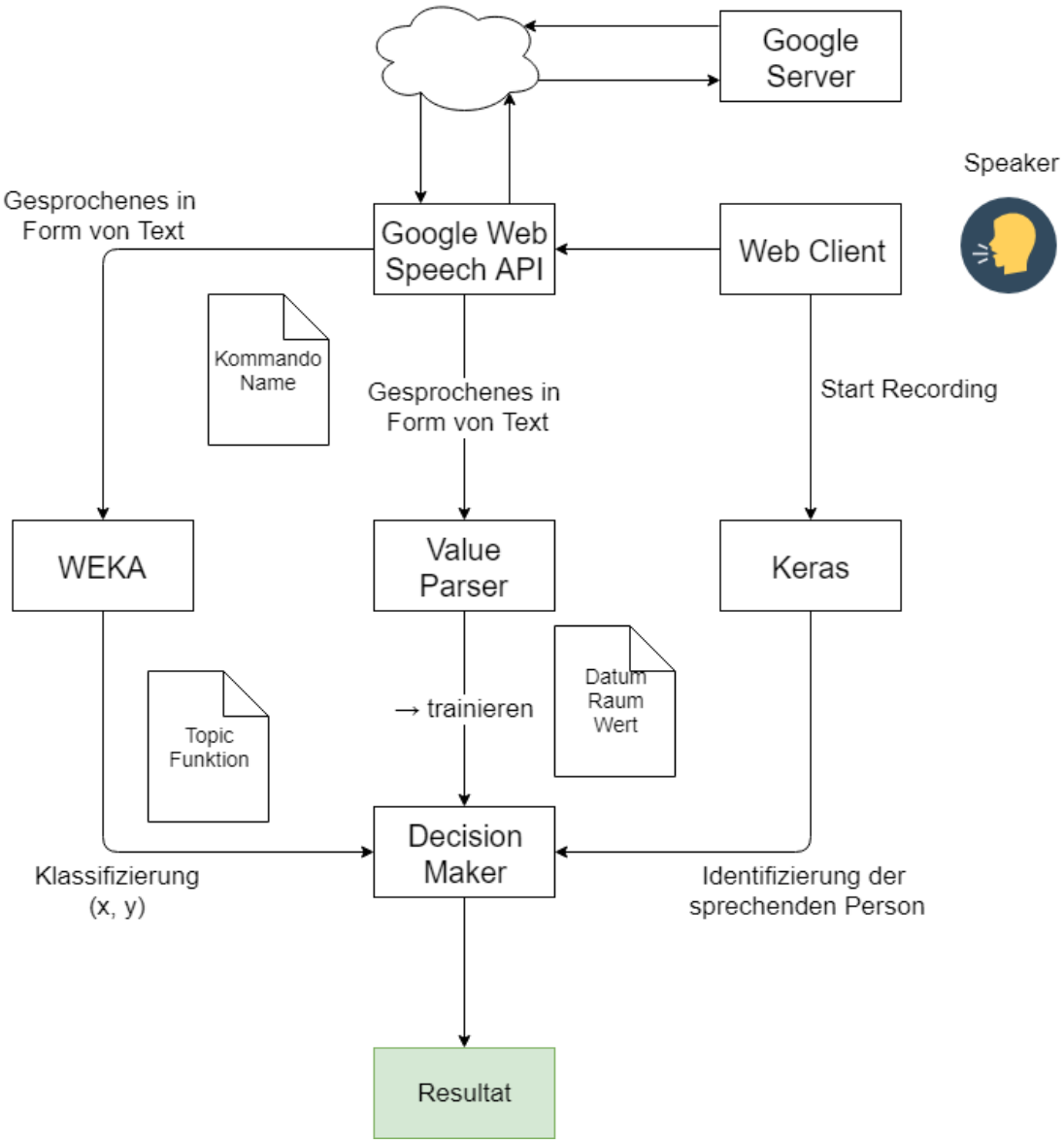


Abbildung 2.3: Prozesskette des Systems

Die Prozesskette zeigt den vereinfachten Ablauf des Systems. Die Kästchen stehen für die einzelnen Komponenten und die Pfeile für die Ergebnisse, die an die nächste Komponente weitergegeben werden.
Anhand desselben Beispielsatzes werden im folgendem Text die einzelnen Teile beschrieben. Dazu wird jedoch nicht auf die Technologien eingegangen und der technische Hin-

tergrund wird nicht beachtet. Es wird gezeigt, welche Ergebnisse von den einzelnen Komponenten geliefert werden. Damit soll ein kurzer leicht verständlicher Überblick geschaffen werden.
Beispielsatz: Könntest du mir bitte morgen um 17 Uhr das Badezimmer auf 25 Grad vorheizen.

Prozesskette in Weka

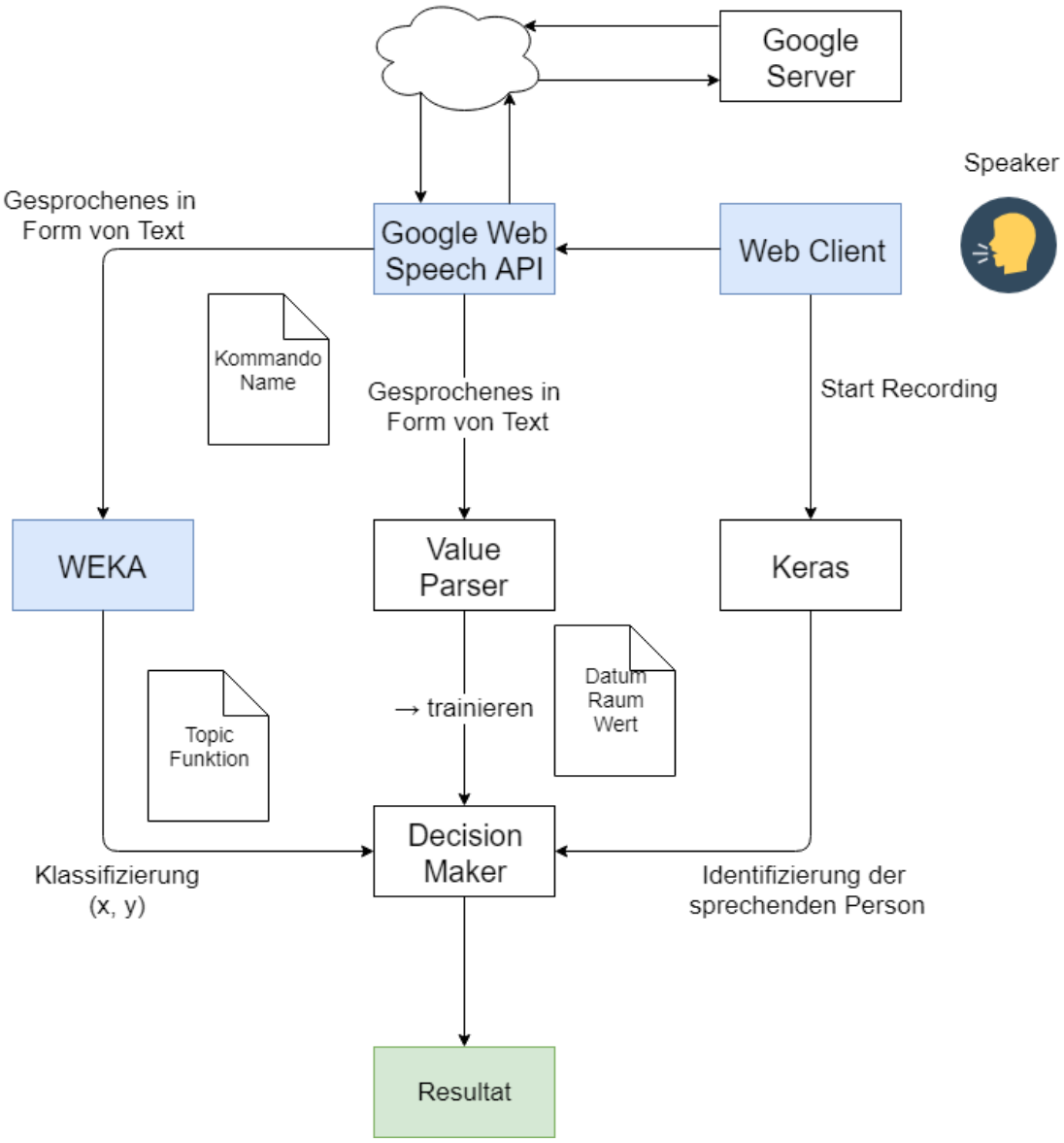


Abbildung 2.4: Prozesskette mit der Sicht auf Weka

1. Der Befehl wird in Form eines Textes von der Google Web Speech API ermittelt und an den Weka REST-Server übertragen der mittels Entscheidungsbäumen den Text bewertet.

2. Der Text wird nun in zwei Kategorien klassifiziert: x und y. Jede Variable nimmt

einen numerischen Wert an. In der ersten Kategorie wird festgestellt, um welches Thema es sich handelt. Beim Beispielsatz will man, dass das Badezimmer vorgeheizt wird. Daraus schließt die Weka Komponente, dass es sich um die Temperatur handelt. Dies bedeutet die Variable x nimmt den Wert 1 an, da die Temperatur so hinterlegt ist. Beispiele für Themen wären:

- Licht = 0
- Temperatur = 1
- Luft = 2

3. Als nächstes wird herausgefunden, was mit x geschehen soll. Im Falle des Beispiels wäre es, die Temperatur erhöhen. Die Temperatur erhöhen bedeutet y nimmt den Wert 0 an. Beispiele für y:

   - Temperatur erhöhen = 0
   - Temperatur verringern = 1

4. Am Schluss wird das Ergebnis [1,0] dem Decision Maker übergeben.

2.2.4 Prozesskette Decision Maker und Value Parser

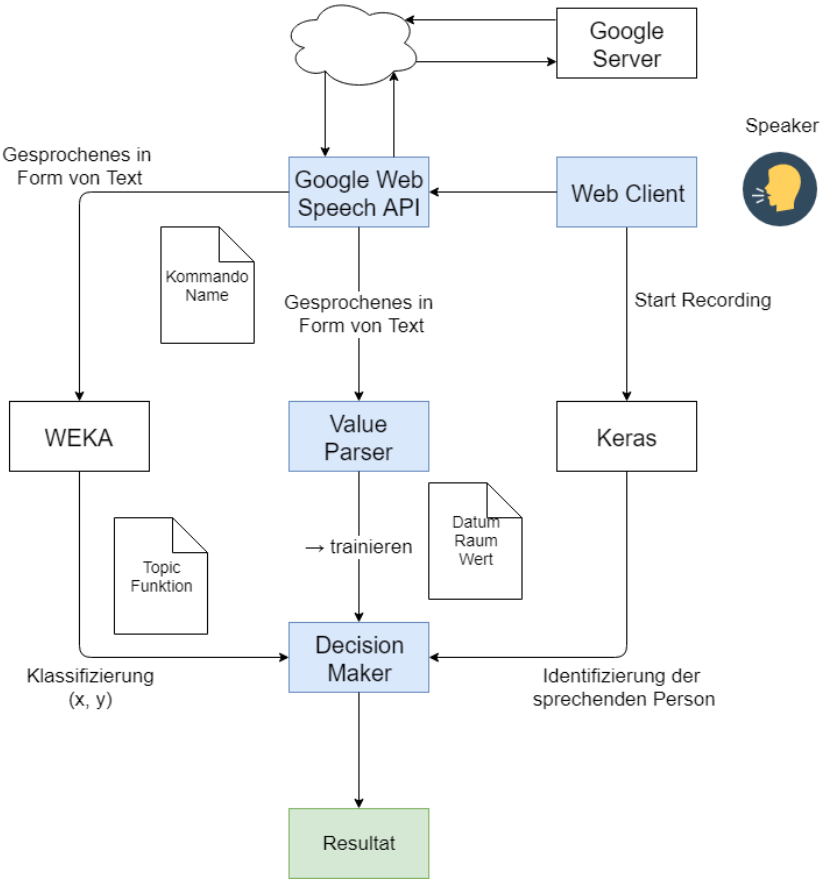


Abbildung 2.5: Prozesskette mit Sicht auf den Decision Maker und Value Parser

1. Die Audiodatei wird an die Google Web Speech API übermittelt. Diese gibt anschließend das Gesprochene als Klartext zurück.
2. Der Valueparser identifiziert die im Text vorkommenden Werte. Diese Werte können sein:
   - skalarer Wert (z. B. 25 Grad)
   - Datum (z. B. in einer Woche)
   - Uhrzeit (z. B. um halb 9)
   - Raum (z. B. Schlafzimmer)
3. Der Decisionmaker erhält von WEKA die Information, in welche Kategorie der Text einzuordnen ist (z. B. Helligkeit, Luftfeuchtigkeit, Temperatur). Somit werden die vom Valueparser erhaltenen Werte zu einem neuen Trainingsdatensatz in der von Weka erhaltenen Kategorie des Decisionmakers.

4. Das Resultat des Decisionmakers ist der Befehl an das Smart Home System.

Beispiel 1 - mit skalarem Wert

Der Sprecher "Max" sagt am 05.02.2018 folgenden Satz: „Könntest du mir bitte morgen um 17:00 Uhr das Badezimmer auf 25 Grad vorheizen."

Vom Valueparser erhaltene Parameter

```
{
"value": "25",
"date": "06.02.2018",
"time": "17:00",
"room": "Badezimmer"
}
```

Von WEKA erhaltene Parameter

```
{
"category": "temperature"
}
```

Von Keras erhaltene Paramete

```
{
"speaker": "Max"
}
```

Es wird der Decision Maker des Benutzers „Max" ausgewählt. Dieser wird daraufhin in der Kategorie „Temperatur" auf die vom Valueparser erhaltenen Werte trainiert.

Das Resultat

```
{
"category": "temperature",
"value": "25",
"date": "06.02.2018",
"time": "17:00",
"speaker": "Max"
}
```

Beispiel 2 - ohne skalarem Wert

Die Sprecherin „Frieda" sagt am 05.02.2017 um 06:00 Uhr im Schlafzimmer folgenden Satz: „Mir ist es zu dunkel." Derzeit is das Licht auf 50 Prozent gedimmt.

Vom Valueparser erhaltene Parameter

```
{
"value": "null",
"date": "05.02.2018",
"time": "06:00",
"room": "Schlafzimmer"
}
```

Von Weka erhaltene Parameter

```
{
"category": "brightness",
"function": "increase"
}
```

Von Keras erhaltene Parameter

```
{
"speaker": "Frieda"
}
```

Es wird der Decision Maker des Benutzers „Frieda“ selektiert. Dieser wird daraufhin in der Kategorie „Helligkeit“ auf die vom Valueparser erhaltenen Werte trainiert. Da der Valueparser keinen skalaren Wert identifiziert hat, wird der Standardwert der Kategorie (aktuelle Helligkeit + 30 Prozent) verwendet. Der Decision Maker wird anschließend mit diesem Wert trainiert.

Das Resultat

```
{
"category": "brightness",
"value": "80",
"date": "05.02.2018",
"time": "06:00",
"speaker": "Frieda"
}
```

2.2.5 Prozesskette Keras

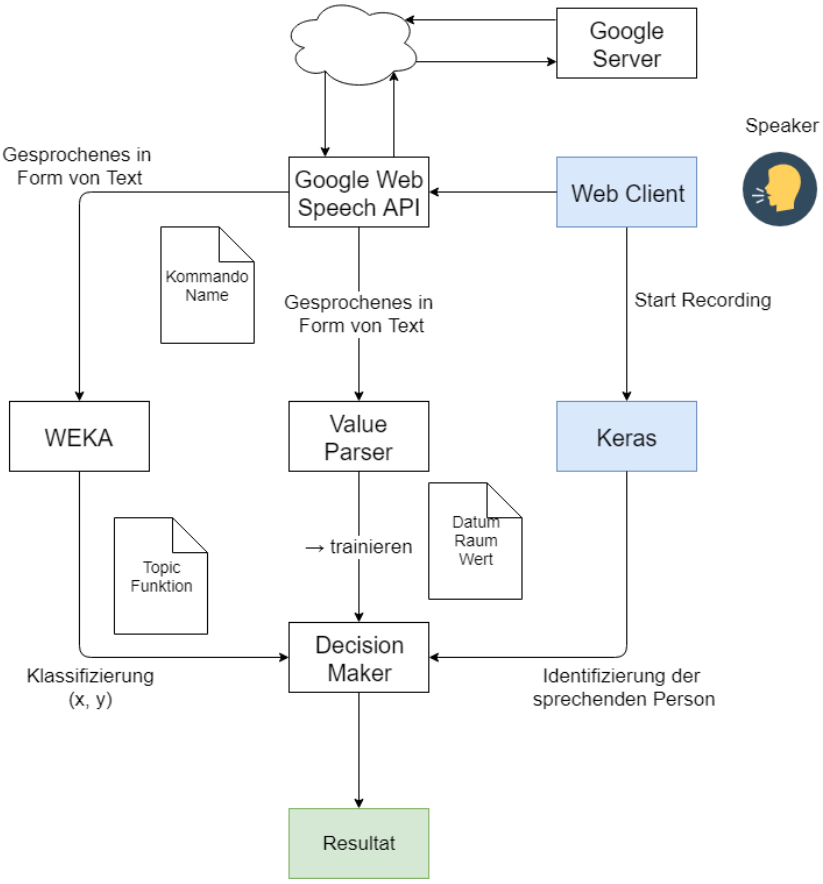


Abbildung 2.6: Prozesskette mit Sicht auf Keras

1. 1. Die Audiodatei wird in das MFCC Format (mel-frequency cepstral coefficients) umgewandelt. Dieses Format eignet sich am Besten für das Erkennen von individuellen Sprachmerkmalen.

2. 2. Die MFCC Daten werden als Parameter an das mit Keras erstellte neuronale Netz übergeben.

3. 3. Das neuronale Netz ermittelt für jeden Benutzer die Wahrscheinlichkeit, dass er oder sie den Satz gesprochen hat.

4. 4. Der Benutzer mit der höchsten Trefferwahrscheinlichkeit wird an den Decisionmaker weitergegeben.

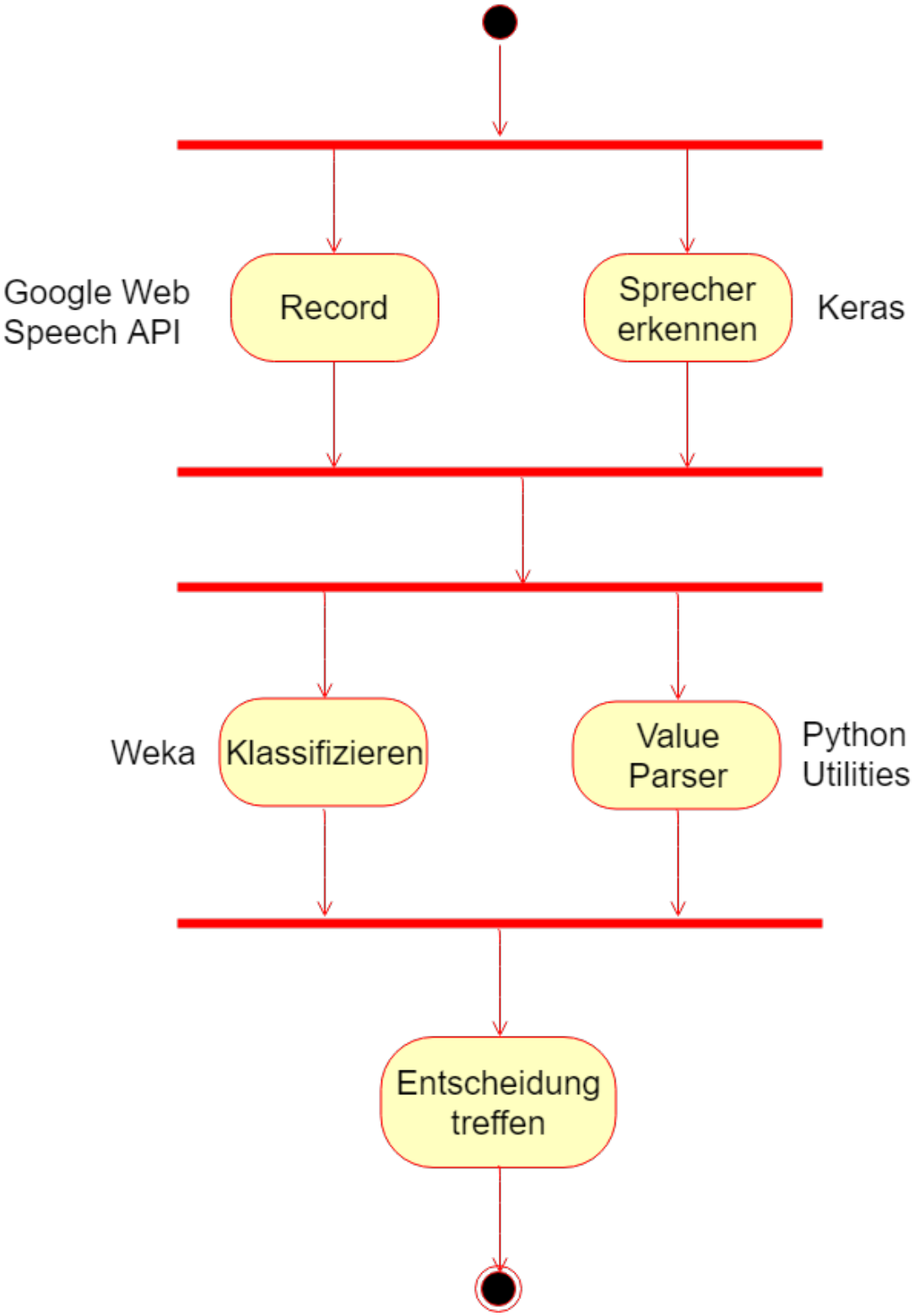


Abbildung 2.7: Aktivitätsdiagramm des Systems

2.2.6 Aktivitätsdiagramm

Mithilfe von Aktivitätsdiagrammen können komplexe Abläufe in einem System modelliert werden. Es werden alle Aktivitäten in einem System mit ihrer zeitlichen Reihenfolge dargestellt. Sie sind hilfreich um sequentielle und parallele Abläufe abzubilden. Parallele Abläufe werden mit Forks begonnen und mit Joins wieder zusammengeführt. Der Start und das Ende werden mit einem Punkt gekennzeichnet. Vergleiche [noaa]
Im Aktivitätsdiagramm (Abbildung 2.7) kann man sehen, dass die Aktivitäten in drei Ebenen unterteilt sind. Als erstes wird über den Web Client anhand der Google Web Speech API das Kommando aufgenommen und in Textform gespeichert. Parallel wird die Stimme von Keras analysiert und der Sprecher wird ermittelt. Sind beide Vorgänge abgeschlossen wird das Ergebnis an den Server gesendet. Dort wird es von Weka klassifiziert und die wichtigsten Werte werden vom Value Parser bestimmt. Anhand aller somit gewonnen Ergebnisse wird am Schluss eine Entscheidung über das endgültige Resultat getroffen.

2.2.7 Deployment Diagramm

Ein Deployment Diagramm (Abbildung 2.8) oder auch Verteilungsdiagramm zeigt welche Hardware Komponenten im System existieren und welche Software darauf läuft. Es wird eine Sicht auf Rechnerknoten, Komponenten und Artefakte (Software) gegeben. Bei der Spoken Command Engine wird als Hardware auf einen Raspberry Pi 3 gesetzt. Am Raspberry ist ein Mikrofon angeschlossen mit dem die Sprachaufnahme über den Angular WebClient, der auf dem Raspberry gehostet wird, getätigt wird. Der aufgenommene Text wird mithilfe der Google Web Speech API in einen String umgewandelt (Speech to Text). Dieser Vorgang findet nicht lokal am Raspberry statt, sondern auf einem Google Web Server. Die Erkennung des Sprechers erfolgt lokal mit dem Keras Flask Server, der auch auf dem Raspberry gehostet wird und auf das selbe Mikrofon wie der Web Client zugreift. Die Interpretation des Textes erfolgt auf einem Java EE Server, der auf einem Wildfly Application Server deployed wird. Dieser wird auch auf dem Raspberry gehostet und speichert die Daten auf eine lokale Derby Datenbank. Vergleiche [noar]

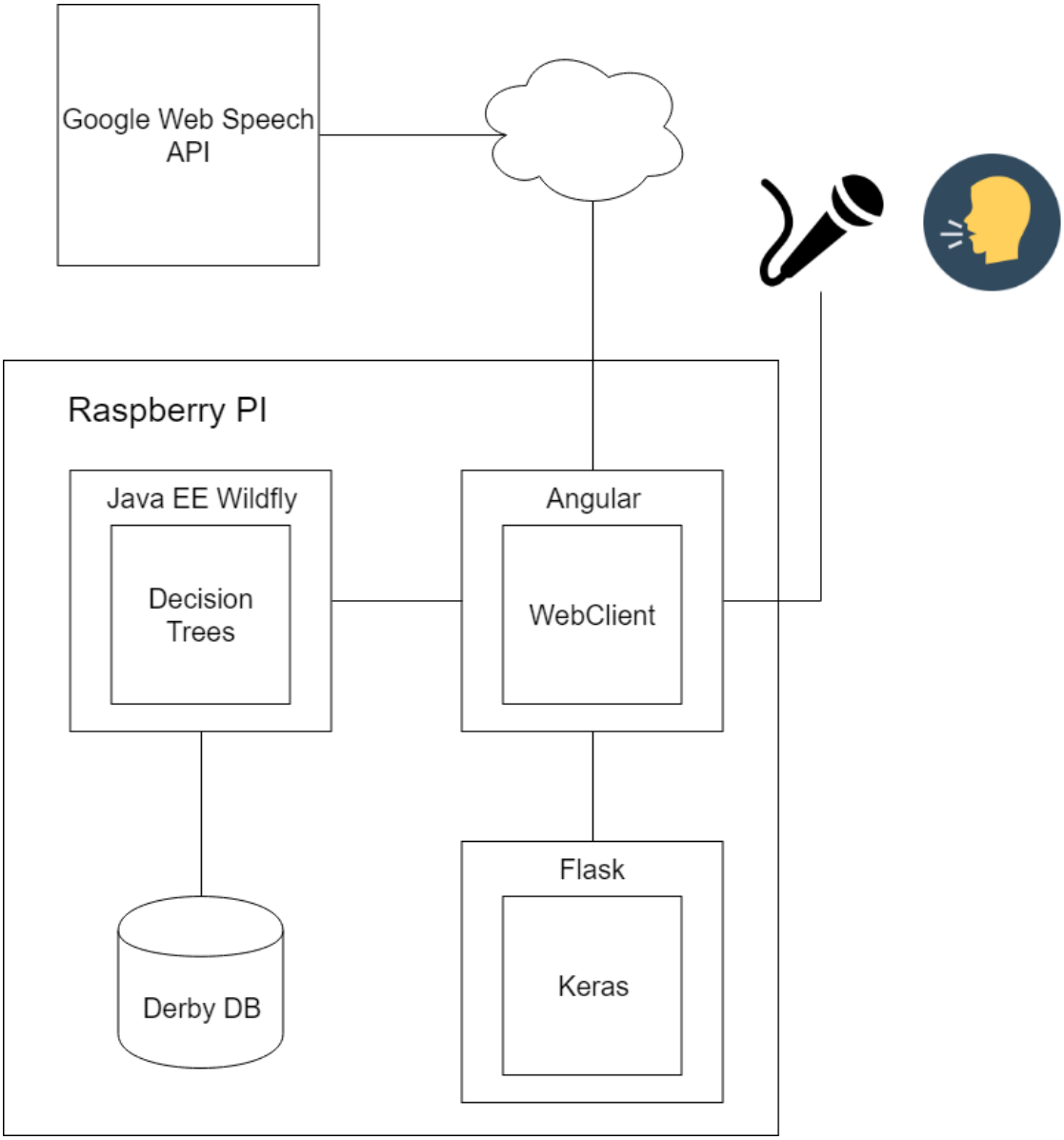


Abbildung 2.8: Deploymentdiagramm des Systems

2.2.8 Klassen Diagramm

Command Facade

In der Klasse Command Facade werden alle Datenbankzugriffe wie persist, update, delete, etc. implementiert. Es wird der Entity Manager injiziert und über ihn wird die Verbindung zur Datenbank hergestellt. Die Command Facade ist ein Singleton. Damit

wird sichergestellt, dass nicht mehrere Verbindungen auf die Datenbank aufgebaut werden, sondern immer die gleiche Instanz verwendet wird. Sie dient auch zur Kapselung vom Rest Endpoint.

Command Endpoint

Die Command Endpoint Klasse bildet alle möglichen REST-Zugriffe des Systems ab. Einmal die Methode um ein Kommando zu interpretieren und um die derzeitigen Präferenzen zu erhalten. Die Datenbankzugriffe erfolgen über die Injektion der Command Facade.

Command

Command ist eine Entitätenklasse, in der das Ergebnis eines interpretierten Befehls gespeichert werden. Es werden alle Klassifizierungen des Entscheidungsbaumes, sowie des Value Parsers und der Spracherkennung festgehalten. Es wird als Entity annotiert, damit es mit der Java Persistence API auf der Datenbank abgespeichert werden kann.

Repository

Im Repository werden alle Entscheidungsbäume und die Python Factory instantiiert. Das Repository ist als Singleton implementiert. Dies ist notwendig um zu gewährleisten, dass jeder Entscheidungsbaum und die Python Factory nur einmal erstellt werden. Wenn sie benötigt werden, wird immer dieselbe Instanz über das Repository aufgerufen.

Python Factory

In der Python Factory wird mithilfe von Jython der Value Parser und der Decision Maker aus Python instantiiert. Die Python Factory wird über das Repository aufgerufen.

Type - Interfaces

Die DecisionMakerType, InputParameterType und ValueParserType Interfaces werden benötigt um die Python Scripts und deren benötigten Methoden in Java abzubilden.

Tree Generator

Der Tree Generator ist eine statische Klasse, in der sich alle Methoden befinden, die zur Generierung und Testung von Entscheidungsbäumen notwendig sind. Außerdem werden hier die ARFF Dateien eingelesen.

Interpretor

Im Interpretor wird mithilfe aller Entscheidungsbäume und dem Value Parser ein Kommando interpretiert und das Ergebnis bestimmt.

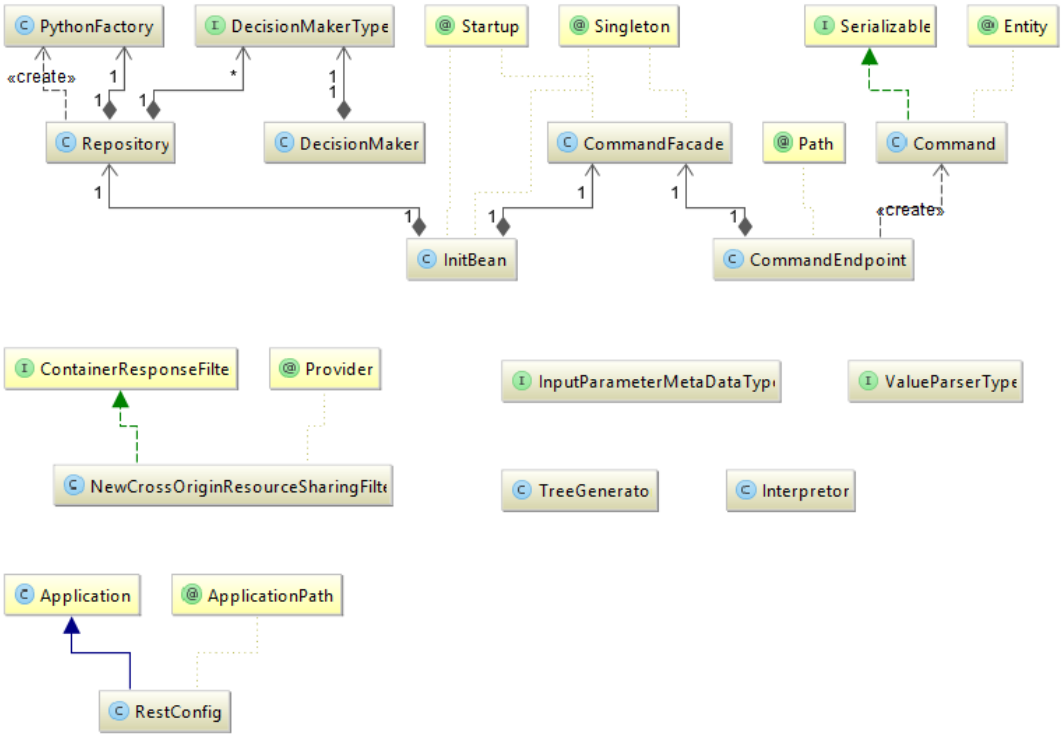


Abbildung 2.9: Klassen Diagramm des Systems

2.3 Implementierung von Weka in Java

2.3.1 Installation

Um das Weka Framework für Java verwenden zu können, muss man entweder die .jar herunterladen und in das Java Projekt einbinden, oder die Dependencies in die pom.xml einbinden. Bei der Implementierung wurde die zweite Variante verwendet. Die Dependencies wurden wie folgt eingebunden:

```
<dependency>
    <groupId>nz.ac.waikato.cms.weka</groupId>
    <artifactId>weka-stable</artifactId>
    <version>3.6.6</version>
</dependency>
```

Abbildung 2.10: Maven dependency von Weka

2.3.2 Implementierung

Einlesen der ARFF Dateien

Um mit dem Weka Framework zu arbeiten, müssen als erstes die ARFF Dateien, aus denen ein Entscheidungsbaum generiert werden soll, eingelesen werden. Dazu wird die Methode getInstances verwendet. Sie hat einen Parameter, den Pfad der Datei als String. Der Rückgabewert ist ein Objekt vom Typ Instance mit dem der Entscheidungsbaum erstellt wird. Wenn die ARFF Datei gefunden wurde und gültig ist, wird die neue Instanz aus den Trainingsdaten erstellt. Als letztes muss noch das Attribut aus der ARFF Datei gesetzt werden, nach dem klassifiziert wird. Dies ist meistens das letzte Attribut eines Datensatzes. 2.11

```
public static Instances getInstances(String path) throws Exception {
    ConverterUtils.DataSource source = new ConverterUtils.DataSource(path);
    Instances newInstances = source.getDataSet();
    if (newInstances.classIndex() == -1){
        newInstances.setClassIndex(newInstances.numAttributes()-1);
    }
    return newInstances;
}
```

Abbildung 2.11: Einlesen der ARFF Datei

Entscheidungsbaum generieren

Die Methode generateTree hat den Parameter data vom Typ Instance. Dies ist das abgespeicherte Modell eines Baumes, der zuvor eingelesen wurde. Mithilfe eines Classifiers kann aus einem Modell ein einsatzbereiter Entscheidungsbaum generiert werden. In diesem Fall wird der Algorithmus J48 für die Generierung verwendet. Nach dem eine Cross Validation auf das Modell durchgeführt wurde und alle Filter gesetzt wurden, wird mit buildClassifier aus dem Modell der Entscheidungsbaum erstellt (Abbildung 2.12). Vergleiche [noad]

```
public static FilteredClassifier generateTree(Instances data) throws Exception {
    FilteredClassifier result = new FilteredClassifier();
    result.setFilter(getFilter(data));
    result.setClassifier(new weka.classifiers.trees.J48());
    evaluateTree(result, data);
    result.buildClassifier(data);
    return result;
}
```

Abbildung 2.12: generieren des Baumes aus der Instanz der ARFF Datei

Modell Filter

In der Methode getFilter werden alle benötigten Maßnahmen getroffen, die für die erfolgreiche Erstellung des Entscheidungsbaumes notwendig sind. Zuerst wird der IteratedLovinsStemmer instanziert und dem Modell zugewiesen. Stemmer sind Alogrithmen im Bereich Computerlinguistik. Sie werden benötigt um Wörter auf ihren Wortstamm zurückzuführen, daher der Ausdruck SStemming". Verkürzungsregeln werden solange auf ein Wort angewandt, bis es die Minimalzahl von Silben aufweist. Danach wird mithilfe des String to Word Vector Filter die sogenannte Tokenization durchgeführt. Ein Satz wird auf seine wichtigsten Eigenschaften untersucht. Wörter wie zum Beispiel: der, in, auf, auch, werden wie "die Spreu vom Weizen"getrennt. Mit den wichtigen Wörtern wird dann der Entscheidungsbaum aufgebaut. Die Methoden setIDFTransform und setTFTransform geben an, nach welchen Berechnungsverfahren vorgegangen wird. Zuletzt wird der Filter zurückgegeben und beim Classifier gesetzt. (Abbildung 2.13) Im folgenden Beispiel ist das Kategorie Model des Entscheidungsbaumes bildlich dargestellt (Abbildung 2.14). Es wurden die wichtigsten Schlüsselwörter gefunden und je nach Vorkommen dieser Wörter wird eine Entscheidung getroffen. Vergleiche [noal] [noan]

```
private static StringToWordVector getFilter(Instances data) throws E
    IteratedLovinsStemmer stemmer = new IteratedLovinsStemmer();
    StringToWordVector filter = new StringToWordVector();
    filter.setInputFormat(data);
    filter.setIDFTransform(true);
    filter.setTFTransform(true);
    filter.setOutputWordCounts(true);
    filter.setStemmer(stemmer);
    return filter;
}
```

Abbildung 2.13: Filter dem Modell zuweisen

```
J48 pruned tree
------------------

Fenster <= 0
|   Luft <= 0
|   |   stickig <= 0
|   |   |   Luftfeuchtigkeit <= 0
|   |   |   |   Temperatur <= 0
|   |   |   |   |   Grad <= 0
|   |   |   |   |   |   kalt <= 0
|   |   |   |   |   |   |   vorheizen <= 0
|   |   |   |   |   |   |   |   waermer <= 0
|   |   |   |   |   |   |   |   |   warm <= 0
|   |   |   |   |   |   |   |   |   |   heiÃŸ <= 0
|   |   |   |   |   |   |   |   |   |   |   kaelter <= 0: 0 (43.0/3.0)
|   |   |   |   |   |   |   |   |   |   |   kaelter > 0: 1 (2.0)
|   |   |   |   |   |   |   |   |   |   heiÃŸ > 0: 1 (2.0)
|   |   |   |   |   |   |   |   |   warm > 0: 1 (3.0)
|   |   |   |   |   |   |   |   waermer > 0: 1 (3.0)
|   |   |   |   |   |   |   vorheizen > 0: 1 (3.0)
|   |   |   |   |   |   kalt > 0: 1 (4.0)
|   |   |   |   |   Grad > 0: 1 (4.0)
|   |   |   |   Temperatur > 0: 1 (9.0)
|   |   |   Luftfeuchtigkeit > 0: 2 (2.0)
|   |   stickig > 0: 2 (3.0)
|   Luft > 0: 2 (5.0)
Fenster > 0: 2 (6.0)
```

Abbildung 2.14: Kategorie Entscheidungsbaummodel bildlich dargestellt

Cross Validation

In der Methode evaluateTree wird der Entscheidungsbaum auf seine Genauigkeit überprüft. Es wird eine sogenannte Cross Validation ausgeführt. Dabei werden alle Trainingsdaten im Modell, ohne deren Lösung, mit dem Classifier getestet und das Ergebnis wird gespeichert. Danach werden die Trainingsdaten mithilfe eines Zufallsgenerators in eine neue Reihenfolge gebracht. Jetzt wird noch einmal klassifiziert. Die Ergebnisse können in der Konsole angezeigt werden. (Abbildung 2.15) Dies ist ein Beispiel einer solchen Cross Validation. (Abbildung 2.16) Hier sieht man wie der Entscheidungsbaum Kategorie mit den Trainingsdaten getestet wird. Das Ergebnis ist die prozentuale Anzahl der richtigen Klassifizierungen.

```
public static void evaluateTree(FilteredClassifier tree, Instances trainData)
    Evaluation eval = new Evaluation(trainData);
    eval.crossValidateModel(tree, trainData, 10, new Debug.Random(1));
}
```

Abbildung 2.15: Cross Validation auf den Classifier ausführen

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          70               78.6517 %
Incorrectly Classified Instances        19               21.3483 %
Kappa statistic                          0.6526
Mean absolute error                      0.1241
Root mean squared error                  0.3043
Relative absolute error                 38.6859 %
Root relative squared error             76.2599 %
Total Number of Instances               89
```

Abbildung 2.16: Cross Validation Beispiel

Entscheidungsbaum verwenden

In der Methode testTree wird ein neues Kommando mithilfe des entsprechenden Entscheidungsbaumes klassifiziert. Die Methode hat zwei Parameter. Den Baum, nach dem klassifiziert werden soll und das neue Kommando. Das Kommando wird auf eine ARFF Datei mit den benötigten Relationen und Attributen mit der Methode writeCommand geschrieben. Danach wird dem Baum diese Datei übergeben. Zurückgegeben wird das Ergebnis in Form eines Integer. Beispielweise 0, 1, 2, 3. (Abbildung 2.17)

```
public static int testTree(FilteredClassifier tree, String command)
    Repository.getInstance().writeCommand(command);
    Instances testData = getInstances(path);
    for (int i = 0; i < testData.numInstances(); i++){
        double label = tree.classifyInstance(testData.instance(i));
        testData.instance(i).setClassValue(label);
    }
    return Integer.valueOf(testData.instance(0).stringValue(1));
}
```

Abbildung 2.17: Entscheidungsbaum mit einem neuen Kommando verwenden

2.4 Value Parser

Das Ziel des Value Parsers ist es, Schlüsselwörter aus dem Befehlstext zu extrahieren. Es können folgende Werte geparsed werden:

- Datum
- Uhrzeit
- Raum
- Skalare Größe bzw. Prozentangabe

Die grundlegende Strategie des Value Parsers ist das Analysieren des Textes mittels Regular Expressions (Regex). Da eine Datums- oder Uhrzeitspezifikation in der deutschen Sprache sehr verschiedenartig ausgedrückt werden kann, wurde dies zu einer weithin komplexen Aufgabe. Des Weiteren muss der Kontext einer Zahl korrekt interpretiert werden.

Hierzu ein Beispielsatz, um eine der vielen Schwierigkeiten in diesem Prozess zu demonstrieren: „Temperatur im Wohnzimmer in 2 Stunden 20 15 Grad.“ In diesem Befehl wird „in 2 Stunden 20“ als „2 Stunden und 20 Minuten“ interpretiert - als skalare Größe ergibt dabei der Ausdruck „15 Grad“.

Heißt der Befehl jedoch „Temperatur im Wohnzimmer in 2 Stunden 20“, so beinhaltet der Ausdruck „in 2 Stunden 20“ nicht die Zeitangabe „in 2 Stunden 20 Minuten“, sondern muss als Zeitangabe (in 2 Stunden) und skalaren Wert (20 Grad) interpretiert werden.

2.4.1 Der Prozess im Überblick

Der Verlauf der Datenextraktion ist in Abbildung 2.18 illustriert.

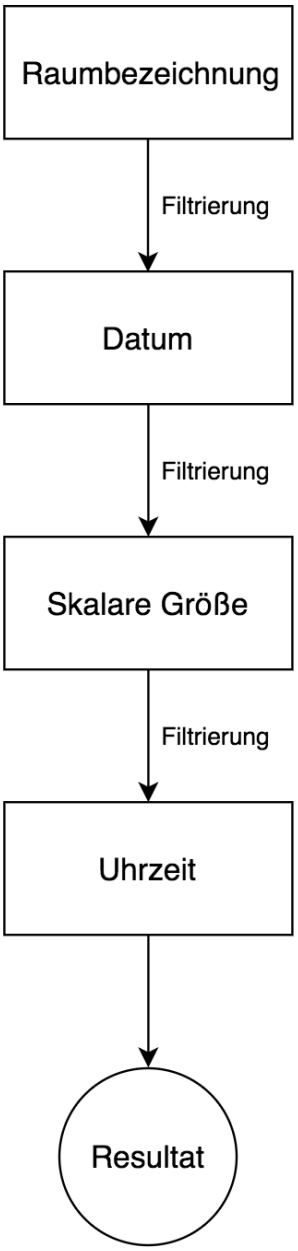


Abbildung 2.18: Reihenfolge der Datenextraktion im Value Parser

1. Überprüfung auf Raumbezeichnung
   - Raumbezeichnung wird notiert und aus dem Text entfernt
2. Überprüfung auf Datumsangabe
   - Datumsangabe wird notiert und aus dem Text entfernt

3. Überprüfung auf skalare Größe
   - Skalare Größe wird notiert und aus dem Text entfernt
4. Überprüfung auf Uhrzeitangabe
   - Uhrzeitangabe wird notiert
5. Alle geparsten Werte werden als Endergebnis zusammengetragen

Beispiel

Um den internen Ablauf zu erklären, wird der Prozess in Abbildung 2.19 anhand eines Beispiels veranschaulicht. Der Beispielsatz lautet: „Übermorgen um halb 6 morgens die Lichtstärke im 2. Stockwerk auf 25 Grad stellen“

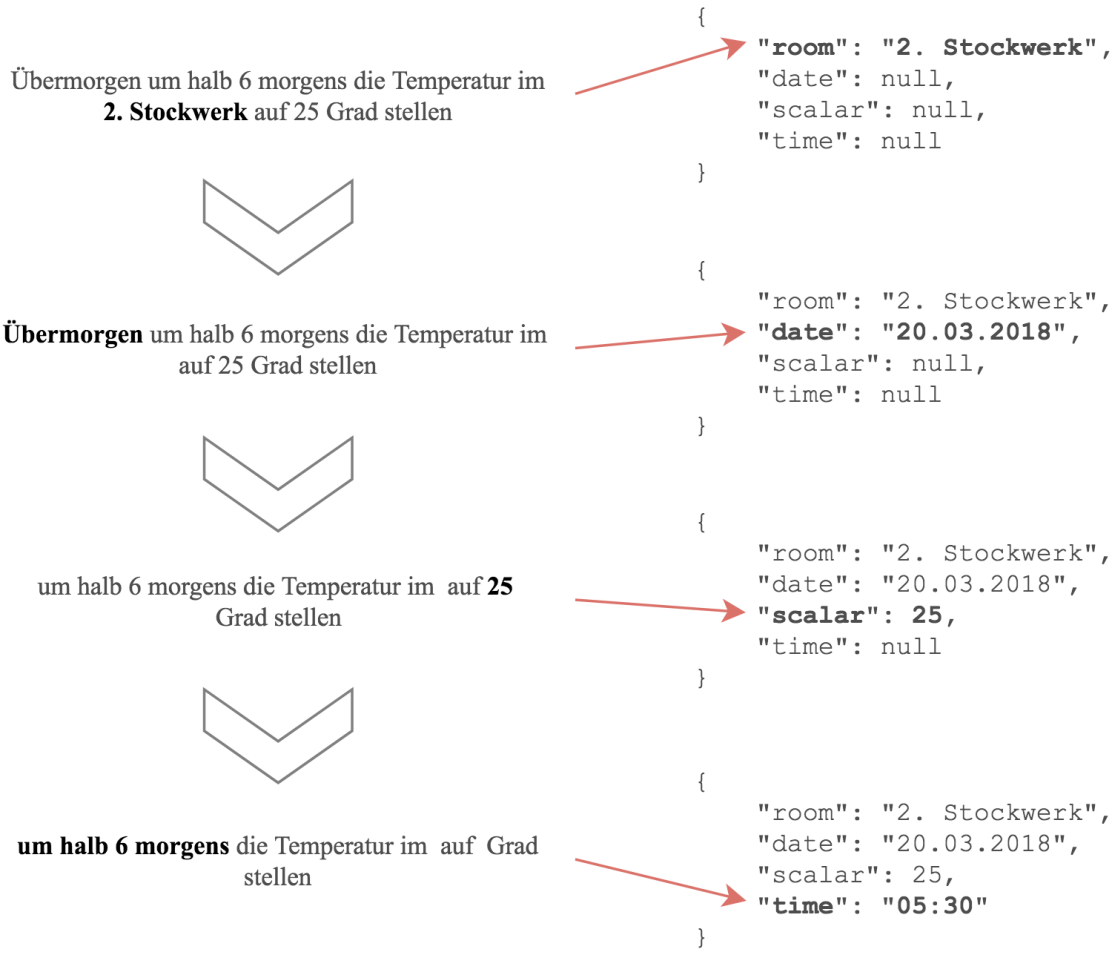


Abbildung 2.19: Ablauf des Value Parsers anhand eines Beispiels

2.4.2 Verwendete Technologien

Der Value Parser wurde ausschließlich in der Programmiersprache Python implementiert. Um das Arbeiten mit komplexen regulären Ausdrücken zu vereinfachen, wurde das Python Modul „re“ aus der Python Standard Library angewendet.

2.5 Decision Maker

Die Aufgabe des Decision Makers ist es, aus den vergangenen Benutzerentscheidungen Schlussfolgerungen über die Präferenzen des Benutzers zu ziehen. Beispielsweise soll anhand der Historie aller Befehle, die die Raumtemperatur betreffen, die aktuell gewünschte Temperatur prognostiziert werden. Der Hintergrund dabei ist, dass sich ein Smart Home System an die Präferenzen des Bewohners dynamisch anpassen soll. Somit ist das System nicht auf eine kontinuierliche Aktualisierung von Seiten des Benutzers angewiesen. Je länger der Decision Maker im Einsatz ist, desto besser kann er die optimale Einstellung vorhersagen und dadurch dem Bewohner den Alltag im eigenen Zuhause erleichtern. In Abbildung 2.20 ist die grundlegende Funktionsweise des Decision Makers am Beispiel der Wärmesteuerung zu sehen.

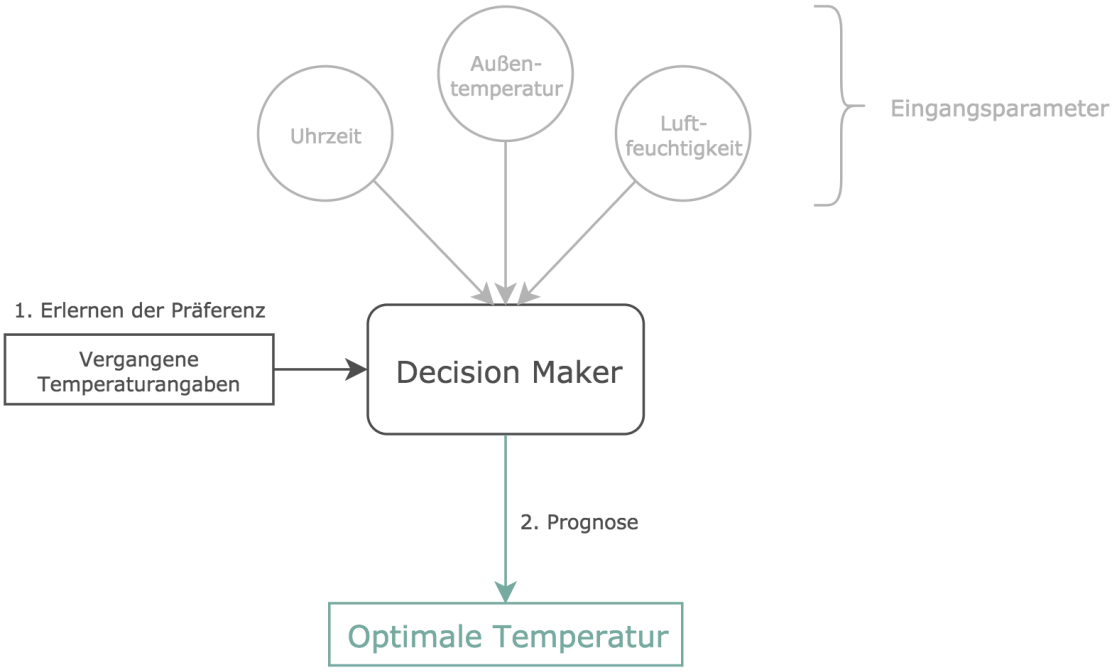


Abbildung 2.20: Die grundlegende Funktionsweise des Decision Makers

2.5.1 Vielseitigkeit in der Anwendung

Wichtig zu erwähnen ist, dass der Decision Maker sehr Abstrakt gehalten ist. Dies ermöglicht den Einsatz nicht nur im Smart Home Bereich, sondern überall dort, wo Zusammenhänge verschiedener Datenmengen erlernt und anschließend daraus Schlussfolgerungen gezogen werden müssen.
Der Einfachheit halber wird in den folgenden Erläuterungen jedoch nur auf die Anwendung im Rahmen dieser Diplomarbeit eingegangen.

2.5.2 Allgemeiner Aufbau

Decision Trend

Die Grundlage des Decision Makers bildet der sogenannte Decision Trend. Dieser stellt die Tendenz in einem bestimmten Wertebereich eines Eingangparameters dar. Folgender Ausdruck zeigt die Berechnung des aktuellen Trends t, wobei n die Gesamtanzahl der erhaltenen Trainingswerte, m die Speicherkapazität, x die Trainingsdaten und ω die Mutabilität repräsentiert:

$$t = \frac{\sum_{i=n-m}^{n} x_i * \sqrt[i]{\omega}}{\sum_{i=n-m}^{n} \sqrt[i]{\omega}} \tag{2.1}$$

Mutabilität Die Mutabilität gibt an, wie veränderlich der Decision Trend ist. Je höher die Mutabilität, desto mehr werden aktuelle Werte in der Berechnung der Prognose bevorzugt. Eine Mutabilität von 0 würde bedeuten, dass der Decision Trend nach einigen Versuchen sehr starr werden würde, da der älteste Datensatz dem Aktuellsten in der Berechnung gleichgestellt ist.

Speicherkapazität Die Speicherkapazität bestimmt, wie weit in die Vergangenheit gegriffen wird, um die Prognose zu ermitteln. Eine Modifizierung dieser Eigenschaft hat ähnliche Auswirkungen auf den Decision Trend wie die Veränderung der Mutabilität. Die Speicherkapazität verpasst den Datensätzen ein Ablaufdatum und die Mutabilität bestimmt, mit welchem Gewicht die von der Speicherkapazität gefilterten Daten in die Prognose einfließen.

Unschärfe der Eingangsparameter Die Unschärfe der Eingangsparameter gibt den Geltungsbereich eines bestimmten Decision Trends bekannt. Im Beispiel der Temperaturregelung repräsentiert der Eingangsparameter eine Uhrzeit. Ist der Initial-Parameter eines Decision Trends 7 Uhr, so ist der Decision Trend bei einer Unschärfe von 2 im Zeitintervall von 5-9 Uhr definiert. Allgemein lässt sich das Geltungsintervall eines Decision Trends folgendermaßen herleiten, wobei p den Initial-Parameter und α die Unschärfe der Eingangsparameter repräsentiert:

$$[p - \alpha | p + \alpha] \tag{2.2}$$

Einfacher Decision Maker

Die elementare Form des Decision Makers ist der Decision Maker mit einem einzigen Eingangsparameter. Hierbei ist das Ergebnis von genau einem Parameter abhängig. Um zum vorherigen Exempel, der Wärmesteuerung, zurückzukehren, wäre dieser Parameter zum Beispiel der aktuelle Zeitpunkt - die gewünschte Temperatur ist somit nur von der momentanen Uhrzeit abhängig.

Im Wesentlichen fungiert ein einfacher Decision Maker als Containerelement für eine Vielzahl an Decision Trends. Dies ist in Abbildung 2.21 am Beispiel der Temperaturregelung veranschaulicht. Die Anzahl der Decision Trends, die ein einfacher Decision Maker beinhaltet, ist abhängig von der Diversität und Vielzahl der vergangenen Eingangsparameter, sowie von der Unschärfe der Decision Trends.

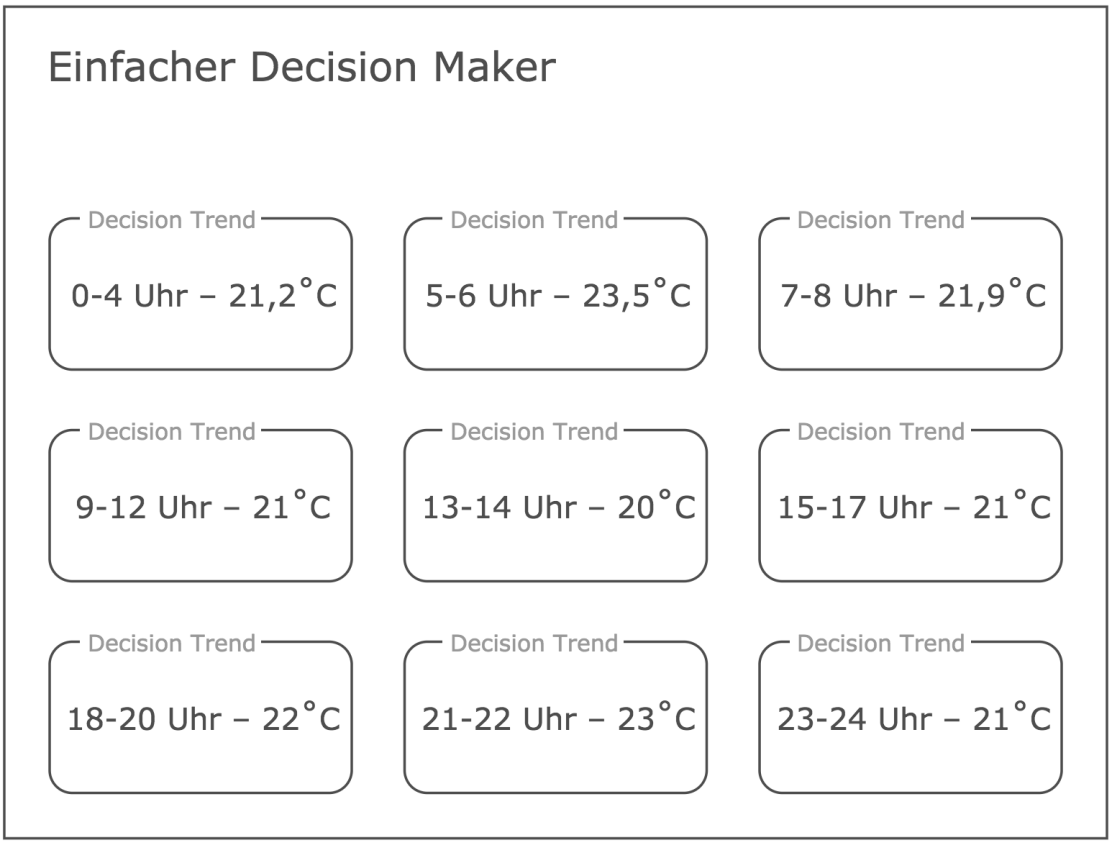


Abbildung 2.21: Einblick in einen einfachen Decision Maker am Beispiel der Temperaturregelung

In Abbildung 2.21 gibt ein einzelner Decision Trend die Temperaturtendenz für ein bestimmtes Zeitintervall (zum Beispiel 0-4 Uhr) bekannt. Je mehr diverse Eingangsdaten hinzukommen, desto kleiner werden die Zeitintervalle eines konkreten Decision Trends - und desto mehr Decision Trends werden benötigt.

Decision Maker mit mehreren Eingangsparametern

In den meisten Anwendungsfällen ist eine Präferenz nicht nur von einer Einflussgröße bestimmt, sondern ergibt sich aus einer Vielzahl an Faktoren. Für diesen Zweck wurde der Decision Maker mit mehreren Eingangsparametern geschaffen. Um am Beispiel der

Wärmesteuerung fortzusetzen, erweitern wir nun den im einfachen Decision Maker bereits eingesetzten Uhrzeitparameter mit der Außentemperatur und der Luftfeuchtigkeit. Der Prozess zur Ermittlung der Temperaturprognose ist in Abbildung 2.22 dargestellt.

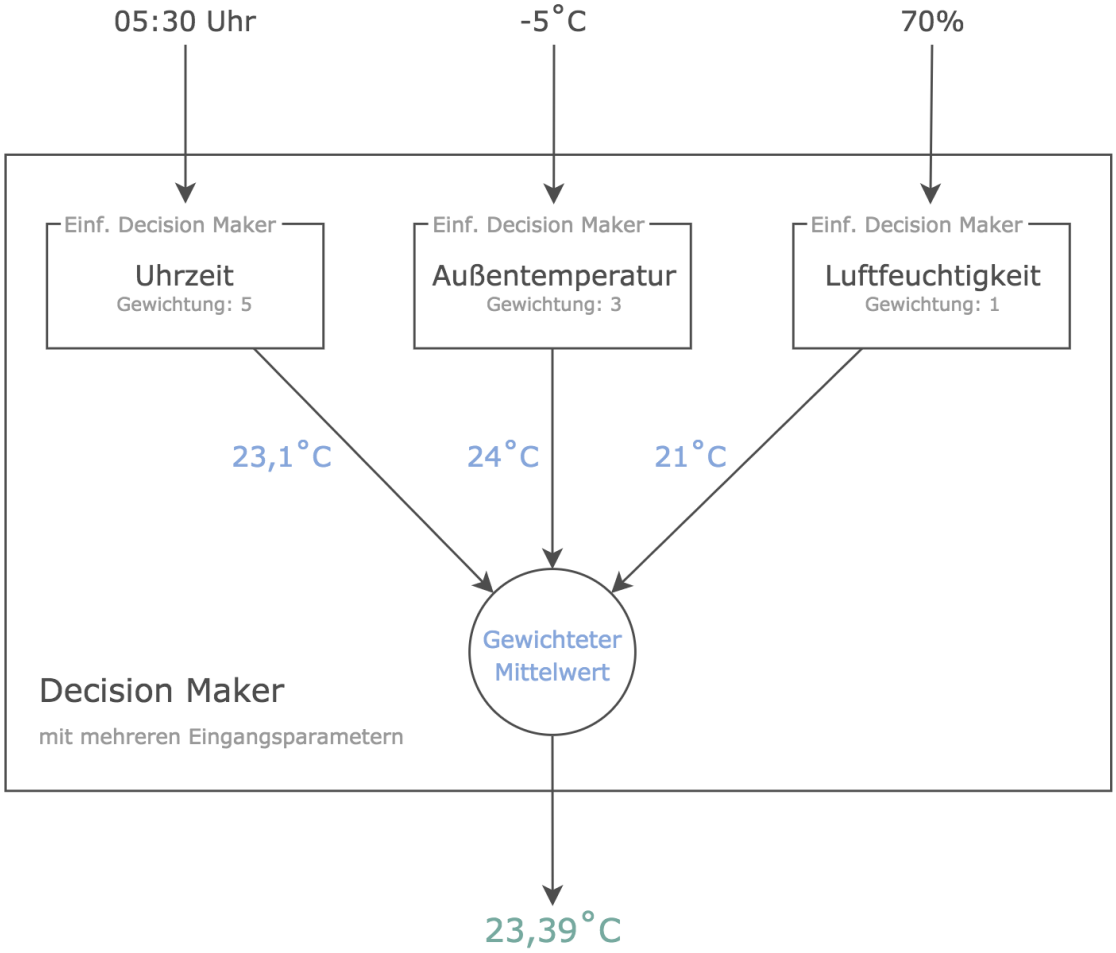


Abbildung 2.22: Interner Ablauf einer Temperaturprognose mit mehreren Eingangsparametern

Für jede Einflussgröße besitzt der Multi-Parameter Decision Maker einen - wie im vorhergehenden Abschnitt beschriebenen - „einfachen Decision Maker“. Diese werden jeweils mit einer Gewichtung versehen, da unter Umständen nicht jeder Parameter gleich hohen Einfluss auf das Endergebnis ausüben soll.

Zusammengefasst erfolgt die Berechnung der optimalen Temperatur durch folgende zwei Schritte:

1. Ermittlung der Temperaturprognosen aller einfachen Decision Maker anhand des jeweiligen Eingangsparameters

2. Berechnung des gewichteten arithmetischen Mittels der Resultate aus 1.

2.6 Speaker Identification

In der Speaker Identification geht es darum, einen Sprecher anhand seiner Stimmmerkmale zu identifizieren. Im Rahmen der Diplomarbeit wurde dazu eine Library namens Keras verwendet, welche auf TensorFlow aufsetzt. Mit diesem Framework können neuronale Netze erstellt, trainiert und getestet werden. Im Falle der Speaker Identification wurde ein rekurrentes neuronales Netz eingesetzt.

2.6.1 TensorFlow

TensorFlow ist eine vom Google Brain-Team stammende KI-Softwarebibliothek. Sie ist sowohl plattformunabhängig als auch öffenlich verfügbar. „Primär verfolgt Google mit TensorFlow zwar das Ziel des ML, jedoch eignet sich das System aufgrund seines allgemeinen Ansatzes und seiner qualitativ hochwertigen Architektur auch für weitere Anwendungsgebiete. [...] Auf der tiefsten Ebene handelt es sich bei TensorFlow um eine Open-Source-Library für nummerische Berechnungen mit Tensoren auf der Basis von Datenflussgraphen.“ (Quelle: web & mobile Developer, Ausgabe 1/18, S. 26)

Tensoren

Wie der Name bereits andeutet, ist der Zentrale Kern von TensorFlow die Durchführung von Berechnungen mit Tensoren. Was sind Tensoren? „In der Mathematik/Physik stellen Tensoren eine Verallgemeinerung des Begriffs der Vektoren und Matrizen dar. Im Unterschied zu Vektoren und Matrizen verfügen Tensoren über beliebige Dimensionen. Ein Tensor lässt sich auch als mathematische Funktion auffassen, die eine bestimmte Anzahl von Vektoren/Matrizen/Tensoren auf eine Zahl/Skalar abbildet.“ (Quelle: web & mobile Developer, Ausgabe 1/18, S. 28)

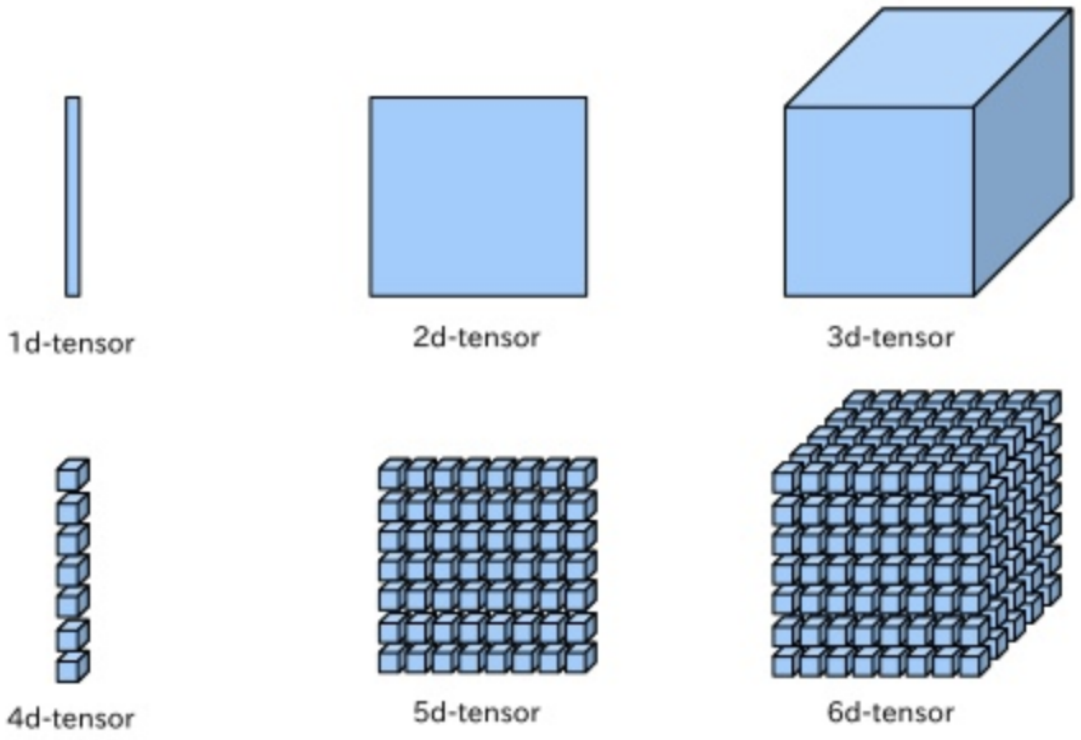


Abbildung 2.23: Dimensionen 1-6 eines Tensors graphisch dargestellt [kma]

In Abbildung 2.23 sind Tensoren der 1. bis 6. Dimension veranschaulicht. Werden Dimensionen 1 bis 3 jeweils als Vektor, Matrix und Quader dargestellt, so kann ein 4-dimensionaler Tensor als Vektor bestehend aus einzelnen Quadern aufgefasst werden. Dementsprechend ist ein 5d-Tensor eine Matrix aus Quadern und ein 6d-Tensor ein Quader aus Quadern. Aus Sicht eines Programmierers sind Tensoren im Grunde multidimensionale Arrays.

Datenflussgraphen

„Datenflussgraphen basieren auf dem Datenflussprinzip: Die Berechnung von Operationen hängt allein von der Verfügbarkeit ihrer Einflussgrößen ab. Die Reihenfolge der Abarbeitung von Instruktionen bestimmt nicht der Kontrollfluss eines Programms, sondern vielmehr die zugrundeliegenden Datenabhängigkeiten.“ (Quelle: web & mobile Developer, Ausgabe 1/18, S. 28)

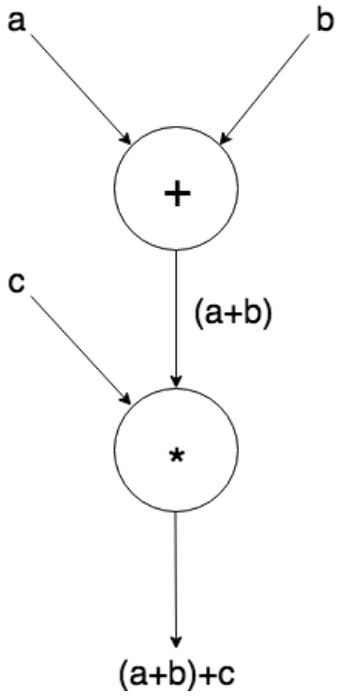


Abbildung 2.24: Datenflussgraph zur Berechnung von (a+b)*c

In Abbildung 2.24 (Quelle: web & mobile Developer, Ausgabe 1/18, S. 28) wird ein einfacher Datenflussgraph dargestellt. Dieser stellt die Rechenoperation (a+b)*c dar. Folgende Schritte sind im Datenflussgraph illustriert:

1. Addition der Summanden a und b
2. Abwarten des Ergebnisses aus Schritt 1
3. Multiplikation der Summe (a+b) mit dem Faktor c
4. Abwarten des Endresultats (a+b)*c

2.6.2 Keras

Keras ist eine Programmierschnittstelle zur Erstellung und Anwendung von neuronalen Netzen, welche in der Programmiersprache Python implementiert ist. Wahlweise kann die zugrundeliegende KI-Softwarebibliothek modular ausgetauscht werden. Hierfür unterstützt Keras folgende Technologien:

- TensorFlow
- CNTK
- Theano

Im Rahmen der Diplomarbeit wurde die von Google entwickelte Open Source Library TensorFlow gewählt. Der Grund hierfür ist, dass TensorFlow nicht nur die aktivste Community besitzt, sondern auch die aktuellere Framework in der vorgenannten Auflistung ist.

Mel Frequency Ceptral Coefficients

Da Audiodateien im WAVE-Format nur eine „[...] zeit- und wertdiskrete Darstellung des zeitlichen Verlaufs eines Signals“ [Wav] enthalten und deshalb zur Stimmanalyse ungeeignet sind, muss das Audiosignal in MFCC umgewandelt werden. „Die Mel Frequency Cepstral Coefficients (MFCC) (dt. Mel-Frequenz-Cepstrum-Koeffizienten) werden zur automatischen Spracherkennung verwendet. Sie führen zu einer kompakten Darstellung des Frequenzspektrums. Das Mel im Namen beschreibt die wahrgenommene Tonhöhe.“ [Mfc]

MFCC Merkmale können in der Struktur als eine Matrix dargestellt werden. Eine Zeile in der Matrix beinhaltet die Merkmale eines bestimmten Zeitabschnittes der Tonaufnahme. Die Zeilen sind nach der zeitlichen Reihenfolge des zugehörigen Audioabschnittes sortiert. Die Spaltenanzahl der Matrix gibt die Anzahl der Merkmale pro Tonabschnitt an. Im Rahmen der Diplomarbeit werden für die Speaker Identification 20 Merkmale pro Audioabschnitt generiert. Folgende Matrix $MFCC$ beschreibt die angewendete Struktur der MFCC Daten, wobei f_{xy} ein Merkmal und t die Anzahl der Audioabschnitte repräsentiert:

$$MFCC := \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,20} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,20} \\ \vdots & \vdots & \ddots & \vdots \\ f_{t,1} & f_{t,2} & \cdots & f_{t,20} \end{bmatrix}$$

Um Störgeräusche außerhalb des Frequenzintervalls der menschlichen Stimme zu eliminieren, wurde ein Bandpassfilter von 50 bis 8000 Hertz angewendet. Für die Berechnung der MFCC Merkmale in diesem Frequenzbereich wurde die Python Bibliothek Librosa verwendet. Im Programmcode sieht dies folgendermaßen aus:

```
wav, sr = librosa.load(file_name, sr=22050)
mfcc = librosa.feature.mfcc(wav, sr, n_mfcc=20, fmin=50, fmax=8000)
```

Zuerst wird das WAVE-Signal mit der Funktion `librosa.load` in die Variable `wav` geladen. Mit dem Parameter `sr` wird die Sample Rate der Signals angegeben. Anschließend werden mit der Funktion `librosa.feature.mfcc` die MFCC Merkmale generiert, wobei folgende Parameter übergeben werden:

1. das WAVE-Signal

2. die Sample Rate
3. die gewünschte Anzahl an MFCC Merkmalen (`n_mfcc`)
4. der Hochpassfilter (`fmin`)
5. der Tiefpassfilter (`fmax`)

Aufbau des neuronalen Netzes

Bei der Identifizierung eines Stimmmusters spielt die Reihenfolge der Sprachabschnitte eine signifikante Rolle. In der Implementierung des neuronalen Netzes wurde daher auf ein rekurrentes neuronales Netz gesetzt, um „zeitlich codierte Informationen in den Daten entdeckt werden“. [Neuc] Als künstliche Neuronen dienen hierbei Long Short-Term Memory (LSTM) Zellen. In Abbildung 2.25 ist der Aufbau des neuronalen Netzes illustriert.

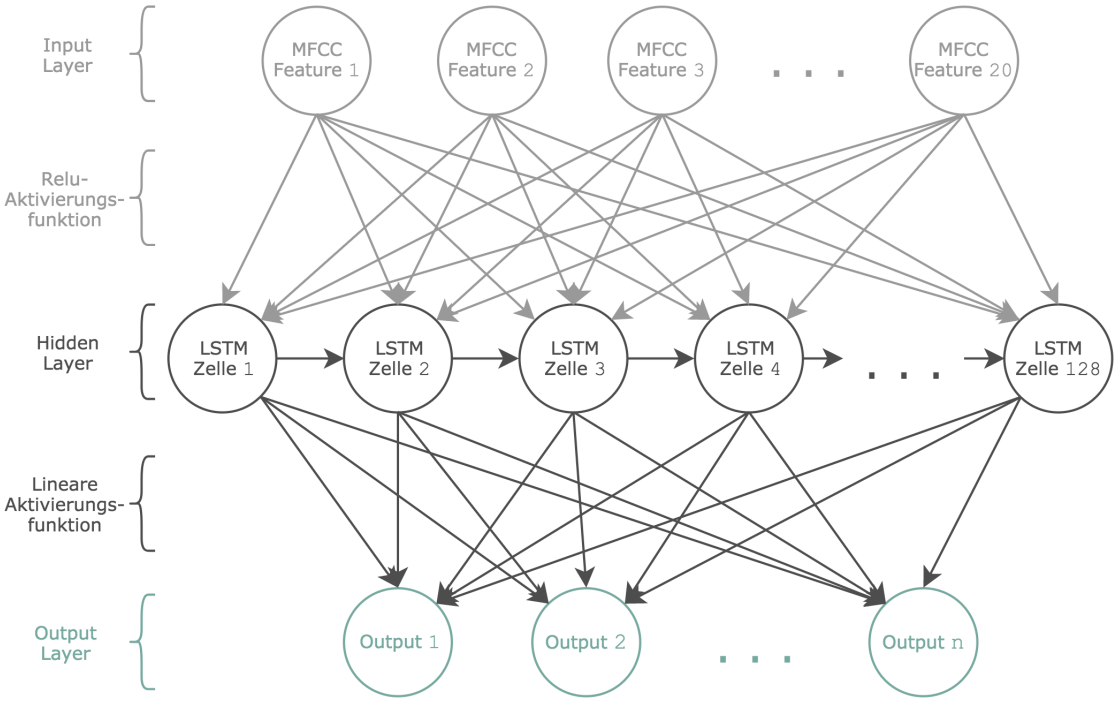


Abbildung 2.25: Aufbau des angewendeten neuronalen Netzes

Folgende Schichten können in der Grafik festgestellt werden:

1. Input Layer mit 20 Eingangsparametern (Anzahl der MFCC Merkmale)
   - Übergang in die nächste Schicht mittels Relu-Aktivierungsfunktion

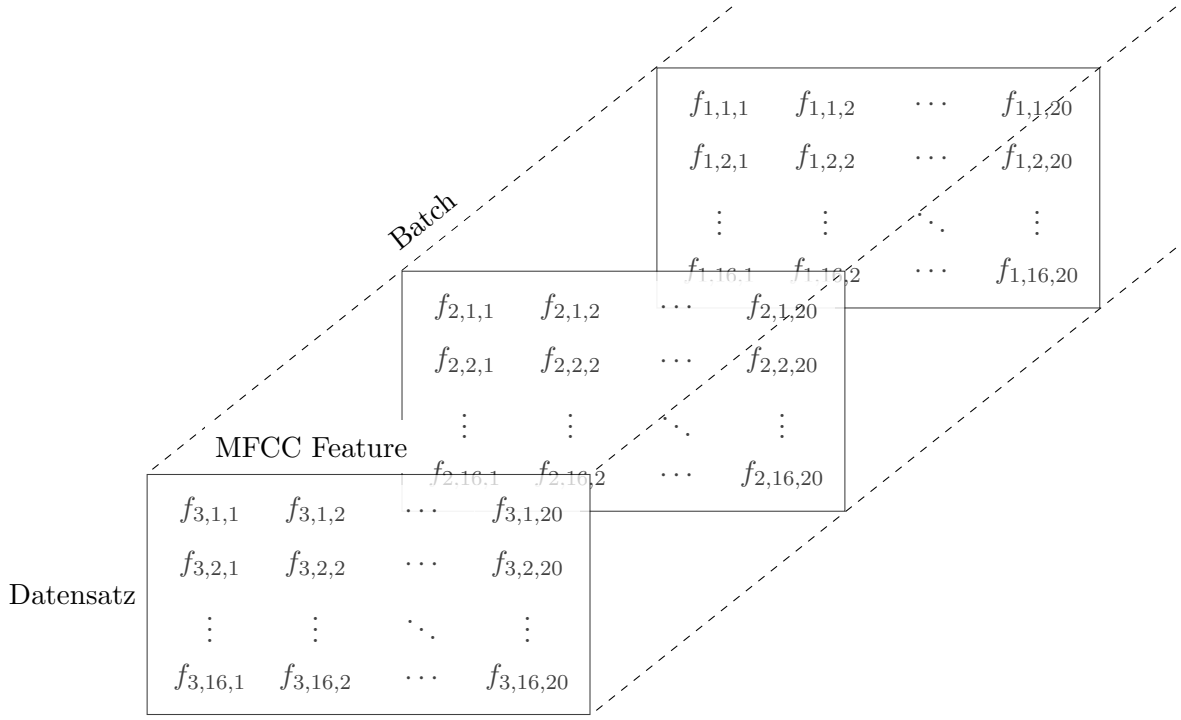
2. Hidden Layer mit 128 LSTM Neuronen - die rekurrente Schicht des Netzes
   - Übergang in die nächste Schicht mittels linearer Aktivierungsfunktion
3. Output Layer mit einer Output-Neurone pro im System enthaltenen Sprecher

Jeder Output-Neurone ist ein Sprecher zugewiesen. Sind in der Speaker Identification beispielsweise 6 Benutzer definiert, so enthält der Output-Layer genau 6 Neuronen und produziert daher genau 6 Ausgabewerte. Der Ausgabewert einer Output-Neurone gibt die Wahrscheinlichkeit an, dass der Sprachabschnitt von dem jeweiligen Sprecher stammt.

Struktur der Trainings- und Testdaten

Wie bereits erwähnt, ist das Hauptmerkmal eines rekurrenten neuronalen Netzes die Fähigkeit, Zusammenhänge aus zeitlich codierten Informationen zu erlernen. Jedoch liefert die Analyse der gesamten Audiosequenz wenig Information über die individuellen Merkmale einer menschlichen Stimme. Der Grund dabei ist, dass sich das Stimmmuster einer Person nicht aus dem Gesamtzusammenhang der MFCC-Daten ergibt, sondern aus der Analyse kleinerer Abschnitte. Deshalb werden die Input-Daten in Batches zu je 16 Datensätzen unterteilt. Nur aus diesen Batches kann das rekurrente Netz ein geeignetes Modell zur Identifizierung eines Sprechers beziehen.

In der folgenden Abbildung werden die ersten drei Batches der hierdurch entstandenen Struktur der Eingangsdaten des neuronalen Netzes illustriert. $f_{x,y,z}$ repräsentiert das z-ste MFCC Feature des y-ten Datensatzes des x-ten Batches.



(2.3)

Implementierung des neuronalen Netzes in Keras

In Keras lässt sich die Erstellung und Konfiguration des neuronalen Netzes folgendermaßen beschreiben:

Instanziierung der `Sequence` Klasse Es wird ein Objekt der Klasse `Sequential` erstellt. Sie dient als Basisklasse zur Errichtung und Anwendung eines neuronalen Netzes und ist außerdem der Container für alle Netzwerkschichten.

```
net = Sequential()
```

Hinzufügen einer Umformatierungsschicht Da aus der Reinforcement Learning Umgebung ein dreidimensionaler Tensor erhalten wird und die LSTM-Schicht jedoch einen zweidimensionalen Tensor benötigt, muss der dreidimensionale Input-Tensor in seine äquivalente zweidimensionale Form gebracht werden. Dies geschieht durch Implementierung einer `Reshape` Schicht. Die Größe der ersten Dimension im resultierenden Tensor gleicht der Batch-Größe, die zweiten Dimension ergibt sich aus der Anzahl an MFCC Merkmalen. Da ein Batch genau 16 MFCC Abschnitte zu je 20 Merkmalen enthält, besitzt der Tensor die Form `(16, 20)`.

```
net.add(Reshape(target_shape=(16, 20), input_shape=(1, 16, 20)))
```

Hinzufügen der rekurrenten Schicht Dem `Sequential` Objekt wird eine Schicht mit 128 LSTM-Neuronen angehängt.

```
net.add(LSTM(128, input_shape=(16, 20)))
```

Hinzufügen der Relu-Aktivierungsfunktion Um die vorhergehende Schicht mit einer Relu-Aktivierungsfunktion auszustatten, wird folgender Ausdruck benötigt:

```
net.add(Activation('relu'))
```

Hinzufügen der Output-Schicht Dem `Sequential` Objekt wird der Output-Layer angehängt. Die Quantität der Neuronen in dieser Schicht ergibt sich aus der Anzahl registrierter Sprecher. Diese wird mit dem Ausdruck `len(self.speakers)` ermittelt.

```
net.add(Dense(len(self.speakers)))
```

Hinzufügen der linearen Aktivierungsfunktion Abschließend wird der Output-Schicht eine lineare Aktivierungsfunktion angehängt.

```
net.add(Activation('linear'))
```

Keras Reinforcement Learning

Um Reinforcement Learning in der Speaker Identification zu implementieren, wurde die Keras Erweiterung keras-rl angewendet. Abbildung 2.26 stellt den Zyklus des Systems dar.

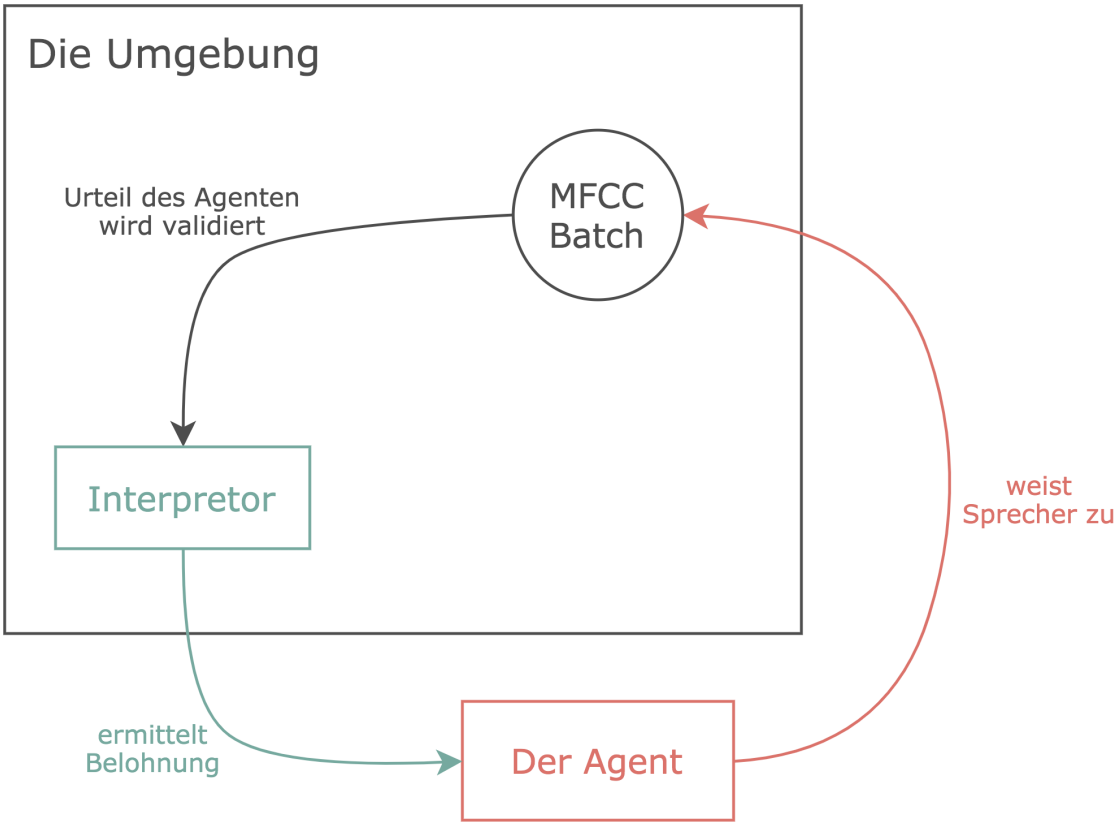


Abbildung 2.26: Reinforcement Learning Umgebung der Speaker Identification

Der Agent Folgendes Codesegment beschreibt die Erstellung und Konfiguration des Agenten. Im Rahmen der Diplomarbeit wurde die Reinforcement Learning Methode „Deep Q-Learning“ angewendet. Keras-rl stellt hierfür die Klasse `DQNAgent` zur Verfügung.

```
memory = SequentialMemory(1800, window_length=1)
policy = BoltzmannQPolicy()

agent = DQNAgent(
    net,
    nb_actions=len(self.speakers),
    memory=memory,
    nb_steps_warmup=10,
    target_model_update=0.01,
    policy=policy
)

agent.compile(Adam(lr=0.001))
```

Zuerst wird das Speichermedium für die Speicherung der Durchläufe erstellt. Der Wert 1800 bezieht sich auf die Größe des Speichers und ergibt sich aus der Anzahl der Trainingsdurchläufe. Anschließend wird die Lernstrategie definiert. Hier kommt die Boltzmann Q-Policy zum Einsatz. Der nächste Schritt ist die Instanziierung des Agenten. Folgende Parameter werden übergeben:

1. das zuvor erstelle neuronale Netz in Form eines `Sequence`-Objekts
2. die Anzahl der möglichen Aktionen (`nb_actions`)
3. das Speichermedium (`memory`)
4. die Anzahl der Aufwärm-Durchläufe (`nb_steps_warmup`)
5. die Aktualisierungsrate des neuronalen Netzes (`target_model_update`)
6. die Lernstrategie (`policy`)

Zuletzt wird der Agent kompiliert. Dabei wird dem Agent die Optimizer-Funktion (Lernregel) übergeben, mit der er das neuronale Netz trainiert. Im Rahmen der Speaker Identification wurde der Adam Optimizer mit einer Lernrate (`lr`) von 0,001 eingesetzt.

Die Umgebung Um mithilfe von keras-rl eine Reinforcement Learning Umgebung zu errichten, muss die vorgegebene Klasse `Env` abgeleitet werden. Dabei ist es erforderlich, die vorgegebenen Methodenköpfe zu implementieren:

- `step(action)`
- `reset()`
- `render(self, mode='human', close=False)`
- `close()`
- `seed(self, seed=None)`
- `configure(self, *args, **kwargs)`

Im Rahmen der Speaker Identification fanden ausschließlich die Methoden `step` und `reset` Anwendung. Der folgende Codeabschnitt zeigt die Implementierung dieser beiden Methoden in der von `Env` abgeleiteten Klasse `SpeechEnvironment`.

```
class SpeechEnvironment(Env):

    def __init__(self, data):
        self.data = data  # beinhaltet alle Trainings-Batches
        self.cursor = 0   # Verweis auf den aktuellen Batch

    def step(self, action):
        reward = 1 if action == np.argmax(self.data[self.cursor][1]) else -1
        self.cursor += 1
        return self.data[self.cursor][0],
               reward, self.cursor < len(self.data), {}

    def reset(self):
        random.shuffle(self.data)
        self.cursor = 0
        return self.data[self.cursor][0]
```

Die Methode `step`: Hier wird die Aktion des Agenten in der laufenden Iteration bewertet und anhand dessen die Belohnung berechnet. Anschließend werden folgende Werte zurückgegeben:

- die aktuelle Observation (der Trainings-Batch des nächsten Durchlaufs)
- die zuvor ermittelte Belohnung der aktuellen Handlung

  bei Richtigkeit: +1

  sonst: -1
- das Abbruchkriterium

Diese Methode wird nach jedem Trainingsdurchlauf von `DQAgent` aufgerufen.

Die Methode `reset`: Hier wird die Reinforcement Learning Umgebung zurückgesetzt, beziehungsweise initialisiert. Im Falle der Speaker Identification werden an dieser Stelle außerdem die Trainingsdaten per Zufallsprinzip durchgemischt, um das neuronale Netz verschiedenen Sortierungen der Daten auszusetzen. Dies verhindert, dass sich die Trainingsdaten in das Netz „einbrennen“. Der Rückgabewert der Methode `reset` ist die Initial-Observation (der erste Trainings-Batch).

Keras im Vergleich zu TFLearn

Anfangs nahm in der Implementierung der Speaker Identification die Softwarebibliothek TFLearn den aktuellen Platz von Keras ein. Im Rahmen der Diplomarbeit wurde anhand mehrere Gründe entschieden, auf Keras umzusteigen. Zum Ersten ist Keras die

von Google offiziell unterstützte high-level Technologie zum Programmieren von neuronalen Netzen. Daraus ergibt sich eine größere Community bei Keras, was sowohl die Entwicklung von Erweiterungsmodulen von Drittparteien fördert, als auch die Problembehandlung von auftretenden Fehlern erleichtert. Außerdem wird bei der Speaker Identification die Keras-Erweiterung keras-rl eingesetzt, welche eine unkomplizierte Erstellung von Reinforcement Learning Systemen ermöglicht.

2.7 Jython

2.7.1 Allgemein

Jython ermöglicht es Python Scripts in Java zu implementieren um ihre Funktionen im Java Programm zu verwenden. Damit kann auf jeder Java Plattform Python Code ausgeführt werden. Mithilfe eines eigenen Compilers kann Python Code direkt auf Java Byte Code kompiliert werden und auf der Java Virtual Machine ausgeführt werden. Dies ermöglicht es, wissenschaftliche Erweiterungen wie NumPy oder SciPy in Java zu verwenden. Java ist im industriellen Bereich weiter verbreitet, Python dagegen im wissenschaftlichen. Jython ermöglicht es beide Vorzüge zu genießen. Vergleiche [noai] [noaj]

2.7.2 Installation

Um Jython verwenden zu können muss Python installiert werden. Je nach Anwendungsfall muss die richtige Python Version ausgewählt werden (meist 3.6). Jython ist ab der Java Version 7 und aufwärts verfügbar. Unter der Verwendung von Maven muss einfach die Jython Dependency in das pom.xml File eingetragen werden.

```
<dependency>
    <groupId>org.python</groupId>
    <artifactId>jython-standalone</artifactId>
    <version>2.7.1b2</version>
</dependency>
```

Abbildung 2.27: Dependencies die für Jython benötigt werden

2.7.3 Verwendung

Python Script in Java abbilden

Um ein Python Script und dessen Methoden in Java zu verwenden, muss man das Script als Java Interface abbilden. Hierbei werden nur die Methoden abgebildet, die auch wirklich verwendet werden. Hilfsmethoden, die nur von den Methoden selbst gebraucht werden, sollen vernachlässigt werden. Die Namen der Methoden müssen im Interface und im Python Script gleich sein. Die Übergabewerte werden auch vom Python Code übernommen.

```
public interface DecisionMakerType {

    void feed_externel_decision(int[] decisionParameter, float outcome);
    void train(int[][] decisionParameter, float[] outcome);
    float decide(int[] parameter);

}
```

Abbildung 2.28: Java Interface, das das Python Script abbildet

Implementierung im Python Script

Damit das Python Script das Java Interface erkennt, muss es wie eine Python Bibliothek importiert werden. Es wird in der Form from [package] import [interface] hinzugefügt. Der [package] Platzhalter ist das package in dem sich das Java Interface befindet. Das [interface] ist der Name des Java Interfaces. In beiden Fällen muss auf die Groß.- Kleinschreibung geachtet werden. (Abbildung 2.29) Zusätzlich wird noch das Java Interface neben der Klassen Definition der Python Klasse angegeben. (Abbildung 2.30)

```
# -*- coding: UTF-8 -*-
from pyinterfaces import DecisionMakerType, InputParameterMetaDataType
import datetime
```

Abbildung 2.29: Importieren des Java Interfaces im Python Script

```
class MultiDecisionMaker(DecisionMakerType):
```

Abbildung 2.30: Java Interface der Python Klasse übergeben

PyObjects erstellen

Auf die Methoden der Python Klassen wird mit sogenannten PyObjects zugegriffen. Ein PyObject wird erstellt, indem mithilfe der PythonInterpreter Klasse die benötigte Python Klasse geladen wird. Es wird dazu die get Methode des PythonInterpreters mit dem Namen der Python Klasse aufgerufen. Dieser Vorgang wird für alle benötigten Python Klassen wiederholt (Abbildung 2.31).

```
public PythonFactory() {
    PythonInterpreter interpreter = new PythonInterpreter();

    interpreter.execfile(DECISION_MAKER_PATH);
    multiDecisionClass = interpreter.get("MultiDecisionMaker");
    inputParameterMetaDataClass = interpreter.get("InputParameterMetaData");

    interpreter.execfile(VALUE_PARSER_PATH);
    valueParserClass = interpreter.get("ValueParser");
}
```

Abbildung 2.31: PyObjects erstellen

Value Parser erstellen

Im folgenden Code wird nun die Klasse ValueParser in Java als PyObject instantiiert. Es müssen nun alle Parameter, die im Konstruktor der Python Klasse benötigt werden, übergeben werden (Abbildung 2.33). Hierbei dürfen jedoch nicht Standard Java Datentypen mitgegeben werden, sondern Jython Datentypen (Abbildung 2.32). Mithilfe der Jython Datentypen weiß das Python Script welche Datentypen im übergeben wurden. In diesem Falle müssen Java booleans zu PyBoolean und Java Listen zu PyListen konvertiert werden. Danach wird das PyObject zu einem ValueParserType explizit umgewandelt.

```
public ValueParserType createValueParser(boolean parseDateTime, boolean parseScalar,
                                         boolean parseLocation, List<String> locationNames){
    PyObject vPObject = valueParserClass.__call__(new PyBoolean(parseDateTime),
            new PyBoolean(parseScalar),
            new PyBoolean(parseLocation), new PyList(locationNames));
    return (ValueParserType)vPObject.__tojava__(ValueParserType.class);
}
```

Abbildung 2.32: Value Parser erstellen

```
def __init__(self, parse_datetime=True, parse_scalar=False,
 parse_location=False, location_names=None):
```

Abbildung 2.33:

Value Parser verwenden

```
PyDictionary dictionary = parser.parse(Repository.getInstance().changeUmlauts(command.getText()));
```

Abbildung 2.34:

2.8 Web Client

Die Spracheingabe eines Befehls erfolgt über einen Angular Web Client. Für die Aufnahme der Sprache wird ein Mikrofon, das am Endgerät angeschlossen ist, verwendet. Das Gesprochene wird mithilfe der Google Web Speech API in einen String umgewandelt. Hierbei wird die Audio Datei an einen Google Server gesendet und ein Text in Form eines Strings wird zurückgegeben. Gleichzeitig wird an den Keras Flask Server die Audiodatei gesendet und sie wird analysiert. Der Sprecher wird an den Angular Client zurückgesendet. Das Kommando in String Form und der Sprecher wird im JSON Format an den Java EE Rest Server gesendet. Hat der Rest Server den Befehl interpretiert, wird das Ergebnis wieder zurück zum Web Client gesendet und dem Benutzer am Browser angezeigt. Vergleiche [noab]

Web Speech API

Die Google Web Speech API ist ein "Injectable Angular Service". Diesen muss man nicht selbst implementieren, sondern nur einige Konstanten auf eigene Bedürfnisse konfigurieren.

- speechRecognition.continues: Die Variable gibt an, ob mehrere Teilergebnisse oder ein einzelnes Gesamtergebnis zurückgegeben werden soll.
- speechRecognition.lang: Gibt die Sprache an, in der die Aufnahme statt gefunden hat. Dann werden die Grammatikregeln der ausgewählten Sprache angewendet, um das Gesprochene in einen Text umzuwandeln.
- speechRecognition.maxAlternatives: Gibt an, wie viele mögliche Ergebnisse pro Wort zurückgegeben werden. Wird zum Beispiel 2 gesetzt, wird das beste und das zweitbeste Ergebnis geliefert. Dann kann selbst entschieden werden welches ausgewählt wird.

```
this.speechRecognition.continuous = false;
this.speechRecognition.lang = 'de-DE';
this.speechRecognition.maxAlternatives = 1;
```

Abbildung 2.35: Web Speech API Konstanten

Der Speech Recognition Service, der zuvor in das Projekt eingefügt wurde, wird nun über den Konstruktor Injiziert. Der Service kann nun jederzeit verwendet werden. Bei

Beendigung des Programmes, wird automatisch die Methode ngDestroy aufgerufen, um das Service Objekt sicher zu löschen.

```
constructor(http: Http, private speechRecognitionService: SpeechRecognitionService) {
  this.http = http;
}
```

Abbildung 2.36: Web Speech API Konstanten

Um die Aufnahme zu starten, wird die Methode speechRecognition.record aufgerufen. Dann muss eine Registrierung beim Oservable durchgeführt werden. Ist die Sprache zu Text Konvertierung fertiggestellt, benachrichtigt die speechRecognition automatisch alle Observer. Nun werden die Methoden je nach Rückgabewert aufgerufen.

- Erhält man einen String, wird dieser in der Variable command abgespeichert und in der Konsole ausgegeben.
- Tritt ein Fehler auf, wird dieser in der Konsole ausgegeben. Im Falle, dass keine Sprache erhalten wurde, wird no-speechïn der Konsole angezeigt.
- Zuletzt wird in der Konsole ausgegeben, dass der Aufnahmeprozess beendet wurde.

```
activateSpeechSearchMovie(): void {
  this.sendRecordStartRequest();
  this.speechRecognitionService.record()
    .subscribe(
      //listener
      (value) => {
        this.command = value;
        console.log(value);
      },
      //errror
      (err) => {
        console.log(err);
        if (err.error == 'no-speech') {
          console.log('no speech')
        }
      },
      //completion
      () => {
        console.log('--complete--');
      });
}
```

Abbildung 2.37: Web Speech API Aufruf

Kapitel 3

Anhang

3.1 Protokolle

Datum: 09.01.2018
13:35 - 16:15
Teilnehmer

- Prof. Mag. Dr. Thomas Stütz
- Immanuel Huber
- Tobias Rechberger

An diesem Tag wurden einige Themen für die schriftliche Arbeit festgelegt. Prof. Stütz gab uns einige Ratschläge für das Kapitel Istzustand. Die Bereiche, die abgedeckt werden müssen, sind: die Ausgangssituation, Informationen über die Firma (mittelständiges Unternehmen) und ein kurzer Überblick, der anhand eines Diagrammes geschaffen werden soll. Kurz erwähnte Prof. Stütz noch, dass wir uns über den Sollzustand informieren sollen und gab uns zwei Dokumente. Einmal über das Pflichtenheft und über das Projekthandbuch.

Datum: 24.01.2018
13:35 - 15:50
Teilnehmer

- Prof. Mag. Dr. Thomas Stütz
- Immanuel Huber
- Tobias Rechberger

Prof. Stütz zeigte uns, wie wir richtig zitieren. Dazu verwendeten wir Zitierregeln von der Technischen Universität Wien. Es wurde auf Mängel im Deployment Diagramm und in der Prozesskette hingewiesen. Weitere Themen, die besprochen wurden, sind: im Sollzustand soll über funktionale - u. nicht funktionale Anforderungen geschrieben werden,

die Prozessketten sollen mit Beispielen genau erklärt werden, keine “wir “ in der schriftlichen Arbeit, im Abstract muss ein Überblick über die gesamte Arbeit gegeben werden, Erklärungen der Technologien sind Theorie und konkrete Anwendung (z.B. Code Beispiele) sind Praxis.

Datum: 08.02.2018
13:55 - 16:20
Teilnehmer

- Prof. Mag. Dr. Thomas Stütz
- Immanuel Huber
- Tobias Rechberger

Am Anfang gab es Probleme den Flask Server mit Websockets zu verwenden. Prof. Stütz legte Wert darauf, dies erneut zu testen, um ein zufriedenstellendes Resultat zu erreichen. Es wurde beschlossen den Value Parser als eigenes Kapitel zu beschreiben, da er doch komplexer war als gedacht. Gemeinsam wurde ein Aktivitätsdiagramm entwickelt, das in die schriftliche Arbeit eingefügt wurde. Am Schluss wurde alles, das bis zu diesem Zeitpunkt geschrieben war kontrolliert.

Literaturverzeichnis

[Bal] Paul Balzer. Neuronale netze einfach erklärt. URL: http://www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert.

[Dur] Murat Durmus. Künstliche intelligenz: Wie maschinelles lernen (machine learning) funktioniert. URL: https://www.aisoma.de/2017/09/25/kuenstliche-intelligenz-wie-maschinelles-lernen-machine-learning-funktioniert/.

[fra] franki. Data-driven market segmentation – more effective marketing to segments using ai. URL: http://www.frankichamaki.com/data-driven-market-segmentation-more-effective-marketing-to-segments-using-ai/.

[Gor] Mandy Goram. Ansätze und vorgehensweisen beim maschinellen lernen. URL: https://www.computerwoche.de/a/ansaetze-und-vorgehensweisen-beim-maschinellen-lernen,3331036.

[Gui] Antoine Guillot. Machine learning explained: supervised learning, unsupervised learning, and reinforcement learning. URL: http://enhancedatascience.com/2017/07/19/machine-learning-explained-supervised-learning-unsupervised-learning-and-reinforc

[Her] Valentin Hermann. Reinforcement learning. URL: http://ekpwww.physik.uni-karlsruhe.de/~tkuhr/Hauptseminar/Hermann_handout.pdf.

[kma] kmario23. How to understand the term ‘tensor‘ in tensorflow? URL: https://stackoverflow.com/questions/37849322/how-to-understand-the-term-tensor-in-tensorflow.

[Knn] Using neural nets to recognize handwritten digits. URL: https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz.

[Kor] Daniil Korbut. Machine learning algorithms: Which one to choose for your problem. URL: https://blog.statsbot.co/machine-learning-algorithms-183cc73197c.

[Mfc] Mel frequency cepstral coefficients. URL: https://de.wikipedia.org/wiki/Mel_Frequency_Cepstral_Coefficients.

[Neua] Neuronale netze - eine einführung - aktivität. URL: http://www.neuronalesnetz.de/aktivitaet.html.

[Neub] Neuronale netze - eine einführung - lernregeln. URL: http://www.neuronalesnetz.de/lernregeln.html.

[Neuc] Neuronale netze - eine einführung - rekurrente netze. URL: http://www.neuronalesnetz.de/rekurrente.html.

[Neud] Neuronale netze - eine einführung - units. URL: http://www.neuronalesnetz.de/units.html.

[Neue] Neuronale netze - eine einführung - verbindungen. URL: http://www.neuronalesnetz.de/verbindungen.html.

[Nie] Michael Nielsen. Using neural nets to recognize handwritten digits. URL: http://neuralnetworksanddeeplearning.com/chap1.html.

[NM] Wolfgang Slany Nysret Musliu. Konzepte der ai: Maschinelles lernen. URL: http://www.dbai.tuwien.ac.at/education/AIKonzepte/Folien/MaschinellesLernen.pdf.

[noaa] Aktivitätsdiagramm. URL: https://www.fbi.h-da.de/labore/case/uml/aktivitaetsdiagramm.html.

[noab] Angular2: Using Speech Recognition of Web Speech API in Angular2 – Muhammad Hassan. URL: https://hassantariqblog.wordpress.com/2016/12/04/angular2-web-speech-api-speech-recognition-in-angular2/.

[noac] Attribute-Relation File Format (ARFF). URL: https://www.cs.waikato.ac.nz/ml/weka/arff.html.

[noad] Cross Validation in Weka - Stack Overflow. URL: https://stackoverflow.com/questions/10437677/cross-validation-in-weka.

[noae] Decision Tree - RapidMiner Documentation. URL: https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/parallel_decision_tree.html.

[noaf] Decision trees explained using Weka | technobium. URL: http://technobium.com/decision-trees-explained-using-weka/.

[noag] Der Aktor | Was sind eigentlich Aktoren oder Aktuatoren? URL: https://www.homeandsmart.de/aktor-aktoren-und-aktuatoren-erklaert.

[noah] HABPanel - UIs - openHAB 2 - Empowering the Smart Home. URL: https://docs.openhab.org/addons/uis/habpanel/readme.html.

[noai] The Jython Project. URL: http://www.jython.org/.

[noaj] Jython – Wikipedia. URL: https://de.wikipedia.org/wiki/Jython.

[noak] KNIME – Wikipedia. URL: https://de.wikipedia.org/wiki/KNIME.

[noal] The Lovins stemming algorithm. URL: http://snowball.tartarus.org/algorithms/lovins/stemmer.html.

[noam] openHAB 2 Documentation - openHAB 2 - Empowering the Smart Home. URL: https://docs.openhab.org/.

[noan] Porter-Stemmer-Algorithmus – Wikipedia. URL: https://de.wikipedia.org/wiki/Porter-Stemmer-Algorithmus.

[noao] RapidMiner – Wikipedia. URL: https://de.wikipedia.org/wiki/RapidMiner.

[noap] Smart Home mit openHAB 2: Installation und Konfiguration - PC-WELT. URL: https://www.pcwelt.de/a/smart-home-mit-openhab-2-installation-und-konfiguration,3443152.

[noaq] Tutorial: Document Classification using WEKA – Karim Ouda – Medium. URL: https://medium.com/@karim_ouda/tutorial-document-classification-using-weka-aa98d5edb6fa.

[noar] Verteilungsdiagramm – Wikipedia. URL: https://de.wikipedia.org/wiki/Verteilungsdiagramm.

[noas] Was ist ein Sensor | Induktiver und kapazitiver Sensor. URL: https://www.homeandsmart.de/sensor-induktiver-und-kapazitiver.

[noat] Was ist ein Smart Home? Geräte, Systeme und Produkte. URL: https://www.homeandsmart.de/was-ist-ein-smart-home.

[Rad] Pranoy Radhakrishnan. Introduction to recurrent neural network. URL: https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3.

[Wav] Riff wave. URL: https://de.wikipedia.org/wiki/RIFF_WAVE.

Abbildungsverzeichnis

1.1 Schwerpunkt liegt in der Interpretation des Kommandos 6
1.2 Unsupervised Learning - das Clustern von Daten [fra] 9
1.3 Klassifizierung und Regression von Beispieldaten [Kor] 10
1.4 Allgemeiner Aufbau eines neuronalen Netzes [Nie] 11
1.5 Schema eines künstlichen Neurons [Knn] 12
1.6 Häufig verwendete Aktivierungsfunktionen [Bal] 13
1.7 Schicht eines rekurrenten neuronalen Netzes [Rad] 14
1.8 Zyklus eines Reinforcement Learning Systems [Gui] 15
1.9 Trainingsdaten für den Entscheidungsbaum 18
1.10 Trainingsdaten abgebildet in Form eines Entscheidungsbaumes 18
1.11 Header Teil des ARFF Files weather 20
1.12 Data Teil des ARFF Files weather 21
1.13 Relation eines ARFF Files 21
1.14 Attribut eines ARFF Files 21
1.15 Nominales Attribut outlook 22
1.16 Deklaration eines String Attributes 22
1.17 Deklaration eines Date Attributes 22
1.18 Deklaration eines Datensatzes mit fehlendem Wert 23
1.19 Schwerpunkt liegt in der Interpretation des Kommandos 25

2.1 Schwerpunkt liegt in der Interpretation des Kommandos 27
2.2 Use Case Diagramm des Systems 29
2.3 Prozesskette des Systems 30
2.4 Prozesskette mit der Sicht auf Weka 32
2.5 Prozesskette mit Sicht auf den Decision Maker und Value Parser 34
2.6 Prozesskette mit Sicht auf Keras 37
2.7 Aktivitätsdiagramm des Systems 38
2.8 Deploymentdiagramm des Systems 40
2.9 Klassen Diagramm des Systems 42
2.10 Maven dependency von Weka 43
2.11 Einlesen der ARFF Datei 43
2.12 generieren des Baumes aus der Instanz der ARFF Datei 44
2.13 Filter dem Modell zuweisen 45

2.14 Kategorie Entscheidungsbaummodel bildlich dargestellt 46
2.15 Cross Validation auf den Classifier ausführen 47
2.16 Cross Validation Beispiel 47
2.17 Entscheidungsbaum mit einem neuen Kommando verwenden 48
2.18 Reihenfolge der Datenextraktion im Value Parser 50
2.19 Ablauf des Value Parsers anhand eines Beispiels 51
2.20 Die grundlegende Funktionsweise des Decision Makers 53
2.21 Einblick in einen einfachen Decision Maker am Beispiel der Temperaturregelung 55
2.22 Interner Ablauf einer Temperaturprognose mit mehreren Eingangsparametern 56
2.23 Dimensionen 1-6 eines Tensors graphisch dargestellt [kma] 59
2.24 Datenflussgraph zur Berechnung von (a+b)*c 60
2.25 Aufbau des angewendeten neuronalen Netzes 62
2.26 Reinforcement Learning Umgebung der Speaker Identification 66
2.27 Dependencies die für Jython benötigt werden 70
2.28 Java Interface, das das Python Script abbildet 71
2.29 Importieren des Java Interfaces im Python Script 71
2.30 Java Interface der Python Klasse übergeben 71
2.31 PyObjects erstellen 72
2.32 Value Parser erstellen 73
2.33 73
2.34 73
2.35 Web Speech API Konstanten 74
2.36 Web Speech API Konstanten 75
2.37 Web Speech API Aufruf 76