HTL Leonding
next level

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING – AUSTRIA

Machine Learning In Microsoft Azure - Einseitige Aufarbeitung

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:
Sljivic Emina

Betreuer:
Karpowicz Michał

Projektpartner:
Bammer Patrick, smartpoint IT consulting GmbH

Leonding, Dezember 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, Dezember 2022 Emina Sljivic

Zur Verbesserung der Lesbarkeit wurde in diesem Dokument auf eine geschlechtsneutrale Ausdrucksweise verzichtet. Alle verwendeten Formulierungen richten sich jedoch an alle Geschlechter.

Abstract

So far, artificial intelligence has only been used sparsely at smartpoint IT consulting. In order to build up a basic knowledge for the company, an independent model was developed in Python by Emina Sljivic as part of this thesis.

The project situation selected was the processing of incoming invoices, where predefined fields were to be extracted. Due to the fact that the data are invoices, it is very important that the model is accurate and reliable.

Since smartpoint specialize in cloud solutions, all methods were developed as Azure Functions, which can be called using an HTTP request. In addition, the implementation was carried out in the Python scripting language, as this allows the model to be used without barriers.

Zusammenfassung

Bislang wurden Künstliche Intelligenzen bei smartpoint IT consulting nur sperrlich verwendet. Um ein Grundwissen für das Unternehmen aufzubauen, wurde im Rahmen dieser Diplomarbeit von Emina Sljivic ein unabhängiges Modell in Python entwickelt.

Als Projektsituation wurde die Verarbeitung von eingehenden Rechnungen ausgewählt, bei denen vordefinierte Felder extrahiert werden sollten. Da es sich bei den Daten um Rechnungen handelt, ist es sehr wichtig, dass das Modell genau und zuverlässig arbeitet.

Da sich smartpoint auf Cloudlösungen spezialisiert, wurden alle Methoden als Azure Functions entwickelt, welche mittels eines HTTP-Requests aufrufbar sind. Zusätzlich wurde hierbei die Implementierung in der Skriptsprache Python durchgeführt, da dies ermöglicht, dass das Modell barrierefrei genutzt werden kann.

Inhaltsverzeichnis

1 Machine Learning - Theorie

1.1 Künstliche Intelligenz

Küntliche Intelligenz (KI) gewann in den letzten Jahrzehnten immer mehr an Beliebtheit und ist im Alltag stets vertreten (auch in Situationen, wo man sie nicht erwartet), jedoch ist diese Art von Technologie keine Neuheit. Erstmals wurde der Begriff KI von McCarthy im Jahr 1956 erwähnt, welcher auf der Dartmouth Universität ein Forschungsprojekt zu diesem Thema führte. [1] Dazu kommen noch Begrifflichkeiten wie Machine Learning (ML) und Deep Learning (DL), welche oftmals als Synonym verwendet werden, allerdings handelt es sich dabei um eigene Bereiche, die sich dennoch in vielen Aspekten überschneiden.

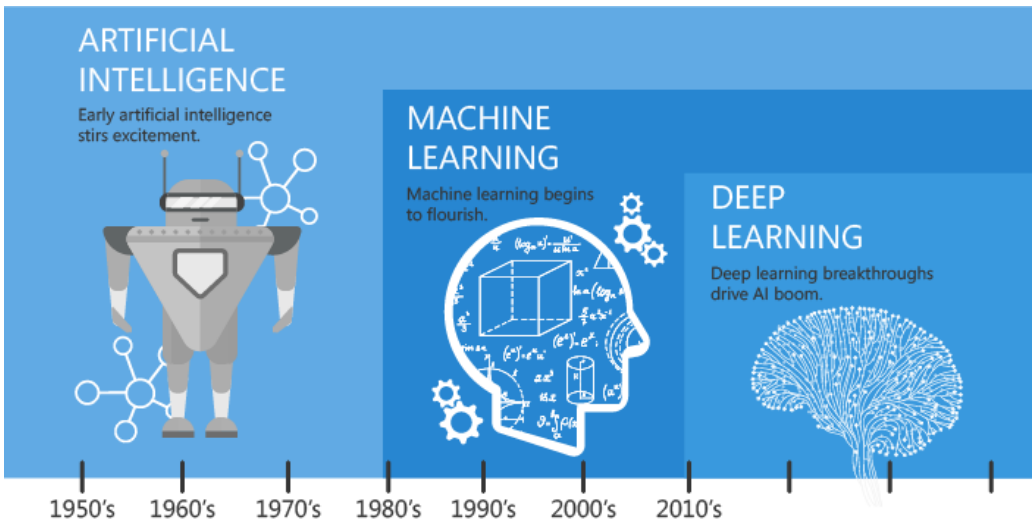

Abbildung 1: Evolution von Künstlicher Intelligenz

Der Begriff KI (Artificial Intelligence (AI)) lässt sich nur schwer genau definieren und umfasst daher ein breites Spektrum der Informatik. Dabei wurden im Verlauf der Geschichte mehrere Definitionen aufgestellt, welche sich auf die unterschiedlichen Interpretationen des Wortes "Intelligenz" fokussieren.

> Artificial intelligence is that activity devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment. [2]

Diese Definition umfasst sowohl komplexe als auch einfach verknüpfte Logiken wie ein Taschenrechner oder vorbestimmte Zuordnungen (Listing 1). Jedoch repräsentieren diese Beispiele nicht den heutigen Fortschritt in diesem Bereich der Technologie.

Listing 1: Zuordnung mit einer if/else-Verzeigung

```
1       if(/* Bedingung */){
2           // Funktionalitaet
3       } else {
4           // Funktionalitaet
5       }
```

Da das Konzept der KI nicht vorgibt, mit welcher Technik ein Problem gelöst wird, bestand die Möglichkeit einer Ausbreitung in allen möglichen Richtungen, welche sowohl Algorithmen als auch Einsatzgebiete inkludieren. Dabei gilt eine Voraussetzung: das Imitieren einer menschlichen Funktion.

1.2 Machine Learning

Bei ML handelt es sich um eine Untergruppe der KI, welche das menschliche Lernen nachahmt. Dabei werden vordefinierte Daten übergeben und die Maschine versucht Strukturen und Muster zu finden, welche dann in Gruppen unterteilt werden. Hierfür ist eine große Menge an Daten nötig, die in der Trainingsphase an das Modell übergeben werden. Im Laufe der Verwendung wird dieses Modell immer präziser, da mit neuen Daten neue Verbindungen geknüpft und vorhandene gestärkt werden können.

Modell Dabei ist ein Modell der Algorithmus, das Muster mithilfe von Daten findet.

Zu den Einsatzmöglichkeiten von ML gehören [3]:

- Spamerkennung von Emails und Telefonanrufen
- Dokumentenklassifizierung
- Chatbots
- Persönliche Assistenten (Siri, Alexa, ...)
- Medizinische Diagnostik
- Betrugserkennung bei Banktransaktionen

1.2.1 Arten

Der wichtigste und komplizierteste Teil beim maschinellen Lernen ist die Kunst, einem Computer das selbstständige Lernen beizubringen, dabei orientiert man sich am Lernprozess eines Menschen. Dazu existieren eine Vielzahl an Ansätzen und zu den bekanntesten gehören:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Um diese Ansätze nachvollziehen zu können, muss man zuerst die menschliche Intelligenz verstehen oder genauer gesagt, die Frage "Wie lernt das menschliche Gehirn?".

Menschliche Ursprünge vom Machine Learning

Bereits als Fötus entwickeln sich Neuronen, die sich über die Zeit verknüpfen und gemeinsam ein Netzwerk bilden, welches dem Körper Anweisungen übermittelt. Daher kommt ein Neugeborenes mit etwa 100 Milliarden Neuronen auf die Welt, die zur Zeit der Geburt nur schwach miteinander verbunden sind. Mithilfe des Lernens werden diese Verbindungen gestärkt, und das Kind kann Vorgehensweisen besser verstehen und neue Erkenntnisse gewinnen, damit steigen zusätzlich auch das Gewicht und die Größe des Gehirnes. [4]

Jedoch verschwinden diese Verbindungen, falls sie nicht aufgefrischt werden und Informationen in Vergessenheit verfallen. Weitere Faktoren könnten der Alterungsprozess oder eine neurologische Krankheit sein, die das Phänomen erklären, dass man im höheren Alter Probleme beim Erlernen von Neuem hat. [5]

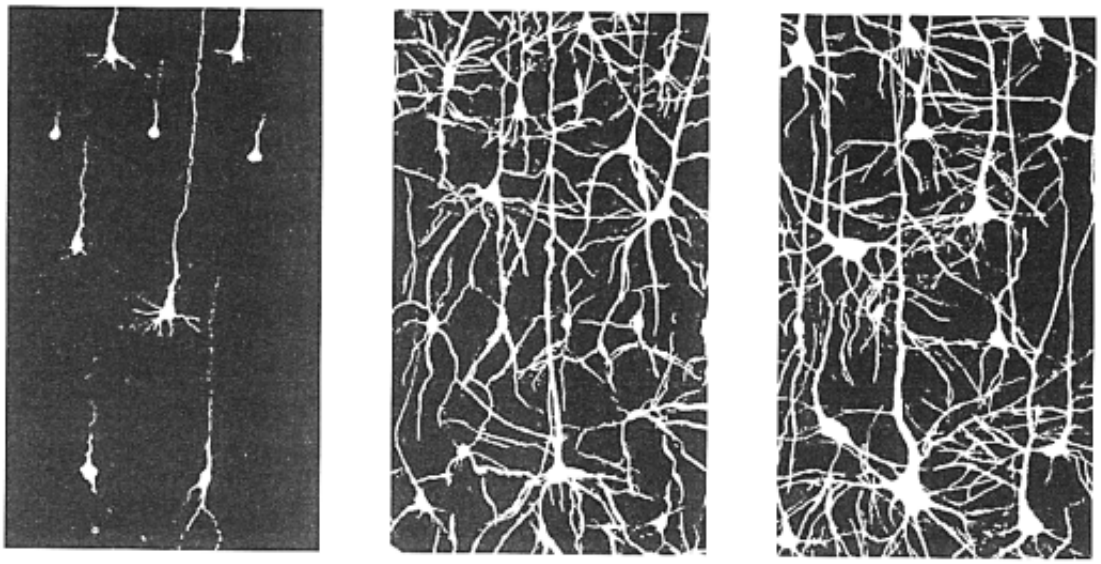

Abbildung 2: Vernetzungen nach der Geburt, nach 3 Monaten und nach 15 Monaten

Neue Informationen verbinden bereits bestehende Neuronen und altes Wissen wird erneuert. Außerdem wird mit oftmaligem Wiederholen das Netzwerk dichter, und man kann das Erlernte leichter abrufen, zugleich werden die neuen Informationen mit dem bereits existierenden Vorwissen besser kombiniert. [4]

Supervised Learning

An einen Algorithmus werden gruppierte Daten übergeben, die neben einer Gruppe noch mehrere Merkmale beinhalten, um danach neue Datensätze zu klassifizieren oder zu regredieren. Die Bezeichnung "Supervised" (im Deutschen "Überwachtes") kommt daher, dass die Gruppen bereits vordefiniert sind und das Programm nicht selber welche erstellen muss. [6]

Farbe	**Form**	**Geschmack**	**Frucht**
rot	herzförmig	süß	Erdbeere
gelb	oval	säuerlich	Zitrone
rot	rund	süß	Apfel
grün	rund	säuerlich	Apfel

Tabelle 1: Beispiel für gruppierte Daten als Tabelle; Merkmale: Farbe, Form, Geschmack; Gruppe: Frucht

Diese Art von Lernen kann man in zwei Typen aufteilen:

- Klassifizierung
- Regression

Klassifizierungs Probleme verwenden Algorithmen, um Daten einer bestimmten Kategorie zuzuteilen. Oft gibt es nur zwei Kategorien wie zum Beispiel Hund/Katze oder Ja/Nein, jedoch gibt es auch Fälle, wo eine Vorhersage mit einer Wahrscheinlichkeit zwischen 0 und 1 getroffen wird. Weiteres gibt es auch Situationen, wo zwischen einer großen Menge an Kategorien ausgewählt wird, zum Beispiel bei der Erkennung von handschriftlichen Ziffern, in diesem Beispiel würde es zehn Möglichkeiten geben.

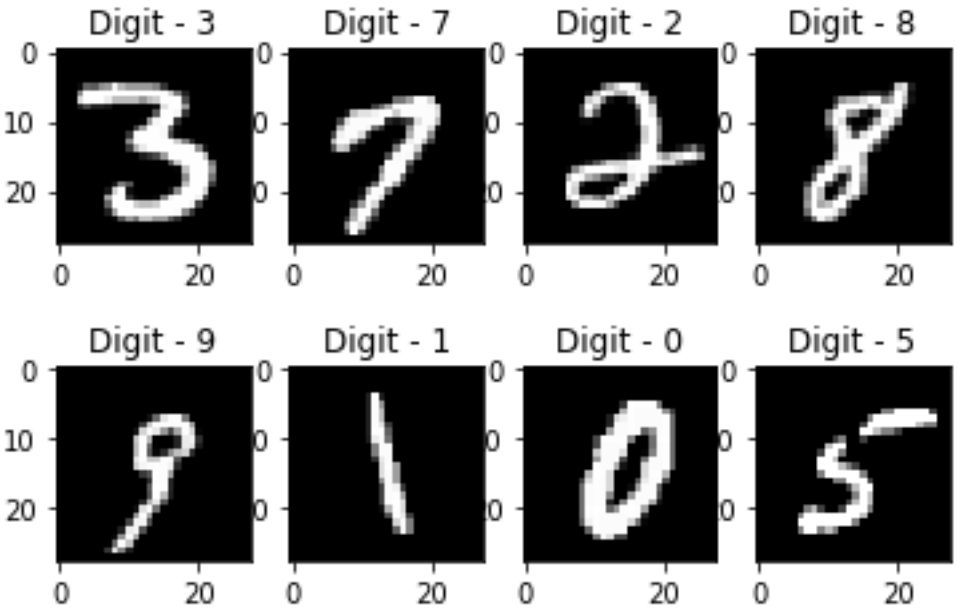

Abbildung 3: Beispiel für eine Klassifizierung von handschriftlichen Ziffern

Zu diesen ”überwachten” Algorithmen gehören Lineare Diskriminanzanalysen, Support Vector Machines (SVM), Random Forests und Entscheidungsbäume. [6]

Entscheidungsbäume (Decision Trees) sind eine hierarchische Abfolge von Bedingungen und sind vergleichbar mit verschachtelten if/else-Statements. Dabei beginnt der Entscheidungsbaum mit einer Bedingung, die auch als ”Root-Node” bezeichnet wird, worauf im Normalfall weitere Bedingungen folgen. Die Blätter dieser Pfade spiegeln die Gruppen oder Klassifizierungen wieder und sind nur über mehrere Pfade erreichbar.

Jedoch haben Entscheidungsbäume das Problem, dass sie sehr gut mit den antrainierten Daten arbeiten und weniger genau mit neuen Daten (Overfitting 1.2.2). Um diese

Genauigkeit zu verbessern, kann man zum Beispiel über Hyperparameter die maximale Länge eines Pfades setzen, wodurch die Bedingungen verallgemeinert werden.

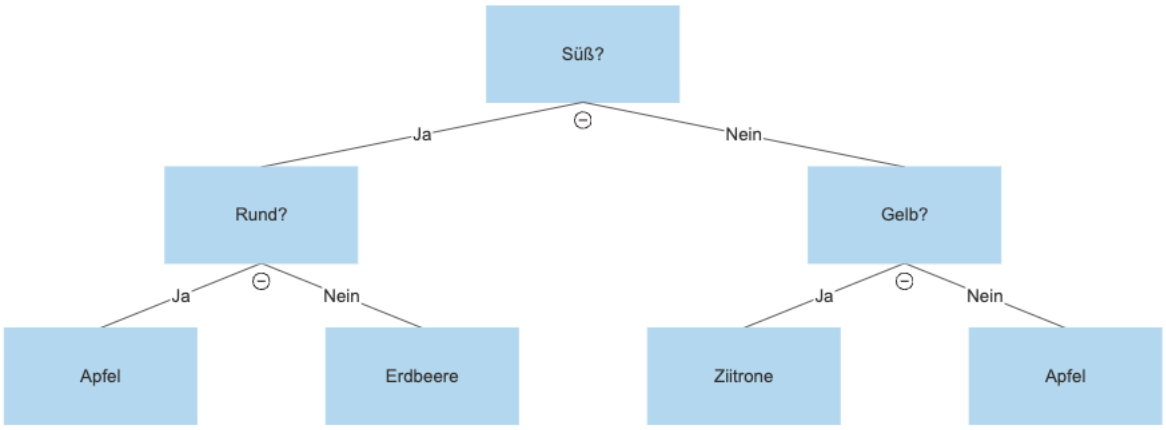

Abbildung 4: Decision Tree an dem Beispiel von Tabelle 1

In diesem Beispiel sieht dieser Entscheidungsbaum noch sehr lesbar aus, jedoch ändert sich dies, wenn Entscheidungen dargestellt werden, bei denen es auf die Nachkommastelle ankommt und wenn die Kategorien sehr schwer differenzierbar sind. Bei dem Supervised Learning erstellt das Programm selbstständig einen Entscheidungsbaum, indem es Muster oder Zusammenhänge findet und analysiert. Nach vielem Lernen kann dieser Entscheidungsbaum optimiert werden und unnötige Verbindungen können entfernt werden.

Fasst man mehrere Entscheidungsbäume zusammen, entsteht ein Random Forest, wobei jeder Entscheidungsbaum nur bestimmte Spalten/Merkmale zugeteilt bekommt. Soll ein neuer Datensatz kategorisiert werden, entscheidet die Mehrheit jener Ergebnisse der Entscheidungsbäume, zu welcher Gruppe dieser Datensatz dazugehört.

Regressionen Unter Regressionen versteht man die Erstellung einer kontinuierlichen Funktion mithilfe von Werten, die auf oder nahe an der Funktion liegen. Sie sind hilfreich, wenn anstatt diskreten kontinuierlichen Werte, wie zum Beispiel die Größe einer Person, festgestellt werden sollen. Dazu gehören lineare Regressionen und polynominale Regression.

Unsupervised Learning

Beim Unsupervised Learning oder unüberwachtes Lernen werden Verbindungen ohne genauere Informationen über den Testdatensatz erzeugt. Dabei muss das Programm

selbst Gruppen definieren und dann die übergebenen Daten in diese Gruppen zuordnen. [6]

Clustering gehört zu den beliebtesten Varianten, um selbstständig Gruppen zu erstellen. Dabei wird jeder Datensatz als Punkt in ein Koordinatensystem mit beliebig vielen Dimensionen eingetragen. Eine Achse stellt ein Attribut dar und je nach Ausprägung ist der Punkt mehr oder weniger vom Ursprung entfernt.

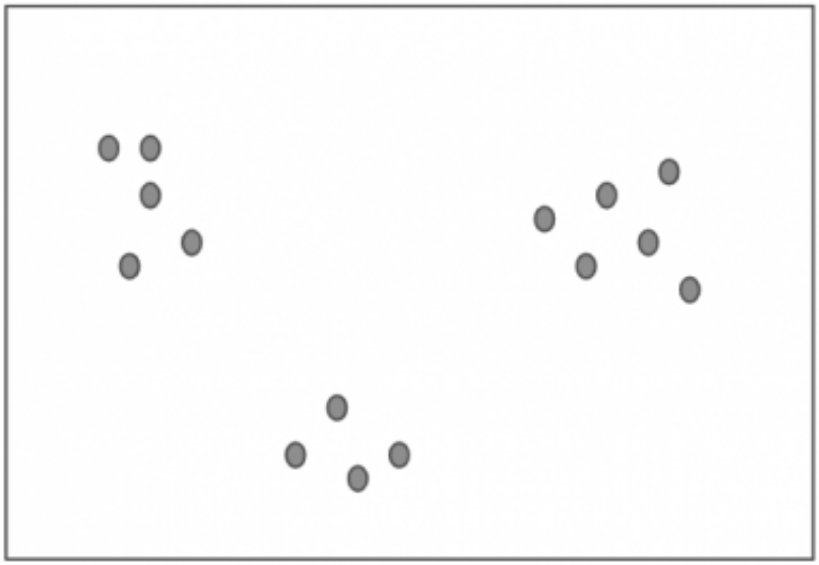

Abbildung 5: Unkategorisierte Daten

Auch für das menschliche Auge ist es möglich, dieses Beispiel (Abbildung 5) in Haufen oder Klumpen zusammenzufassen, genau das gleiche macht ein Programm mit Clustern. Die Interpretation dieser Gruppen muss jedoch wieder durch Menschen erfolgen, da ein Computer nicht im Stande dazu ist, diese Cluster einer Kategorie zuzuordnen.

Diese Vorgehensweise wird oft in sehr komplizierte Einsatzbereiche genutzt und daher ist es sehr schwer, differenzierbare Cluster zu erstellen. Der Prozess, solche Cluster zu definieren, basiert darauf, die Punkte so zu gruppieren, dass der Abstand in diesem Cluster klein und zu anderen Clustern groß ist.

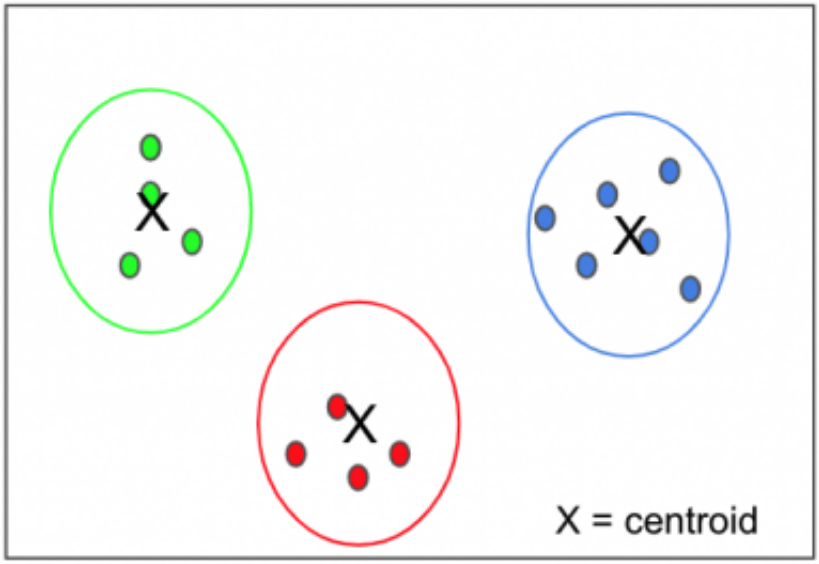

Abbildung 6: Kategorisierte Daten

Reinforcement Learning

Das Reinforcement Learning oder das beschränkte/verstärkte Lernen wird oft mit dem Konzept ”Learning by doing” verglichen, da es sich weniger auf das Ergebnis fokussiert und mehr auf Aktionen oder Vorgängen. Ein Beispiel für diese Vorgehensweise aus der Sicht eines Schülers ist das Üben vor einer Matheschularbeit. Wird während dem Üben ein Fehler gemacht, merkt man sich das Problem und passt sein Verhalten bzw. seinen Rechenweg so an, dass dieser Fehler nicht mehr vorkommt. Diesen Vorgang nennt man auch negative Verstärkung. [6]

Dahingegen führen richtige Ergebnisse oder erwartete Reaktionen zu positiven Verstärkungen und man versucht, dieses Verhalten zu wiederholen.

Durch negative und positive Verstärkungen wird das Verhalten verbessert, um den besten Weg zum Ziel zu finden. Bei komplexen Systemen kann dies selbstständig vom Programm gemacht werden, jedoch bei simpleren kann es geschehen, dass ein unnötig komplizierter Weg gefunden wird. In diesen Fällen schaut ein Supervisor dem Programm ”über die Schulter”.

1.2.2 Probleme

Overfitting

bedeutet, dass ein Modell zu sehr an die Trainingsdaten angepasst ist und auf neue Daten falsche Ergebnisse liefert. Dies hat den Grund, dass das Modell zu lange mit dem

Trainingsdatensatz gearbeitet hat und zu spezifisch wurde. Daher versucht man mithilfe einer Verallgemeinerung über die Hyperparameter das Modell zu vereinfachen.

Underfitting

bedeutet, dass ein Modell viel zu allgemein ist und nicht an die Trainingsdaten angepasst ist. Einerseits könnte es sein, dass das Modell nur kurz mit den Trainingsdaten gearbeitet hat, andererseits kann es sein, dass die Klassifizierung innerhalb des Trainingsdatensatzes eine starke Mehrheit zeigt. [7]

1.2.3 Phasen

Das Erstellen von einem ML-Modell wird in vier Phasen unterteilt, wobei jede Phase eine bestimmte Aufgabe erfüllt.

1. Vorbereitung der Daten / Preprocessing

   Jeder Anfang sollte mit einer Analyse der Daten anfangen. Dabei sollte man die Datentypen, die Anzahl der Daten oder Klassifizierungen und die Anzahl der fehlenden Werte zusammenzufassen und übersichtlich darstellen (Tabellen oder Diagramme). Dies hat zwei wesentliche Gründe und einer davon ist, dass man sich mit den Daten vertraut macht und danach über ein gewisses Verständnis verfügt. Der zweite Grund ist, dass man im Verlauf der Phase die Daten so verarbeiten kann, dass man am Ende das beste Ergebnis erreicht.

   Diese Vorbereitung sollte mit stichprobenartigen Daten angeführt und später durch verschiedene Strategien weitergeführt werden, die dazu dienen, entweder fehlende Daten zu bereinigen oder Ausreißer zu beseitigen.

   Wie man mit fehlenden Daten umgeht, kommt auf die Situation an. Hat man Zugang zu einer großen Menge an Daten, könnte es sinnvoll sein, diese Daten zu löschen. Dabei bleibt die Frage offen, ob man die ganze Spalte (wenn alle fehlenden Werte in einer Spalte sind) löscht oder nur die betroffenen Zeilen. Falls die Entscheidung getroffen wird, dass die Spalte entfernt wird, muss man sich vergewissern, dass damit potenziell wichtige Daten verloren gehen. Ist die Anzahl der Datensätze jedoch klein, hat man die Möglichkeit, fehlende Werte mit dem Mittelwert oder Ähnlichem zu ersetzen, dieser kann aus allen Daten gebildet werden oder aus den Daten, die über dieselbe Klassifizierung verfügen. Sind diese

Mittelwerte nicht repräsentativ, kann dieser durch einen fixen Wert ersetzt werden. Ein Beispiel dafür wäre die Analyse eines Schlaganfall-Datensatzes, in Fällen eines Kindes ist zu vermuten, dass es nicht raucht und daher ist es vertretbar, dass man dafür den Wert 0/false einsetzt. Die letzte Möglichkeit ist, dass man diese Werte mithilfe eines ML-Modells (Regression) bestimmt. [8]

Das Gegenteil der fehlenden Werte sind doppelte Werte, diese können zur Überrepräsentierung von bestimmten Klassifizierungen führen und sollten daher bereinigt werden. Bei ähnlichen Werten ist dies jedoch eine Interpretationssache, da es in manchen Situationen ein unbedachter Messfehler sein könnten. Ein Vorteil dieses Schrittes ist, dass man am Ende von jeder Klasse gleich viele Datensätze hat.

Als Nächstes wäre eine Analyse jeder einzelnen Spalte sinnvoll, anfangend mit den Datentypen. Falls es sich um Enumerationswerte handelt, müssten diese in numerische Werte umgewandelt werden, außer es handelt sich um die Klassifizierungsspalte. Handelt es sich um eine Spalte, wessen Werte eine Rangfolge (z.B.: gut, mittel, schlecht) darstellen, kann man diese mit einer Nummer zwischen 0 und 1 austauschen. Dabei ist zu beachten, dass der minimalste und maximalste Rang entweder den Wert 0 oder 1 zugeteilt bekommen und jeder Rang dazwischen einen Wert dazwischen. Sind es jedoch unabhängige Enumerationswerte, könnte man mithilfe der One-Hot-Encoding Methode die Daten umwandeln, wo jeder Enumerationswerte eine zusätzliche Spalte bekommt und entweder mit 0/1 (false/true) befüllt ist. Außerdem sollten unterschiedliche Einheiten angeglichen werden und jene textuelle Einheit aus dem Wert entfernt werden.

Das letzte Problem sind Ausreißer, um diese zu identifizieren, müsste man die eigentliche Verteilung der Daten kennen. Danach vergleicht man verdächtigte Werte (meistens der größte oder kleinste Wert) mit dem Durchschnitt und entscheidet, ob es sich wirklich um unerklärliche Werte handelt. Diese können mit den gleichen Funktionen wie bei fehlenden Werten ersetzt werden.

2. Visualisierung der Daten

   Die tabellarische Darstellung von Daten erschwert dem menschlichen Auge, Muster zu erkennen, um dies zu umgehen ist jede Art der Visualisierung hilfreich. Damit kann man schnell Insights und Gruppen identifizieren, außerdem kann sie auch bereits bei der Vorbereitung der Daten helfen, da man zum Beispiel mithilfe eines Boxplots Ausreißer deutlich schneller erkennen kann.

Um dies noch leichter zu machen, ist es wichtig, dass die Farbpalette an die gegebene Situation angepasst ist, da es bei schlechter Repräsentation leicht zu Verwirrung kommen kann. Wie zum Beispiel beim Darstellen des Wetters bietet sich die Farbe blau für Regen an und rot/gelb für Sonnenschein.

Jedes Diagramm hat seine Vorteile und Nachteile, wie zum Beispiel beim Violinplot die Verteilung der Daten relativ zur Anzahl der gleichen Klassifizierung sehr gut dargestellt werden. Jedoch hat es den Nachteil, dass diese Darstellungen oft irreführend sind, da sie relativ anstatt absolut sind.

Um Zusammenhänge besser zu erkennen, kann man zwei Variablen in einem Diagramm darstellen.

3. Training des Modells

   Der erste Schritt beim Trainieren ist die Einteilung von Trainingsdaten und Testdaten, hierzu wird eine Einteilung von 70% zum Trainieren und 30% zum Testen angestrebt.

   Es ist wichtig, dass die Testdaten eine ausgeglichene Anzahl von Daten pro Klassifizierung enthalten, da es sonst zu einem unausgewogenen Modell kommen könnte. Wie in 1.2.2 (Overfitting) beschrieben wurde, kann sich ein Modell zu sehr an die übergebenen Trainingsdaten anpassen, sodass später die Testdaten nicht korrekt klassifiziert werden. Um dies vorzubeugen, könnte man einen Teil der Testdaten als Validierungsdaten nutzen, welche die Anpassung der Hyperparameter möglich machen [9].

   Es ist außerdem wichtig, dass die Testdaten lediglich zum Testen benutzt werden, denn sobald das Modell diese Daten verarbeitet, werden sie gespeichert und im späteren Verlauf ebenfalls zur Klassifizierung genutzt.

   Am Anfang sollte man sich für einen traditionellen Algorithmus entscheiden, damit man später jegliche Veränderungen mit einem Basiswert vergleichen kann.

4. Evaluierung und Verbesserung

   Um herauszufinden, ob das antrainierte Modell wirklich nutzvoll und genau ist, können folgende Werte berechnet und verglichen werden: Accuracy (Genauigkeit), Precision (Präzision) oder Recall. Je nach Situation sollte man den Fokus auf einen bestimmten Wert legen. Zum Beispiel beim Testen von Wasserqualitäten würde der wichtigste Wert die Precision sein, denn man kann in Kauf nehmen, dass

nicht schädliches Wasser als schädlich gekennzeichnet wird, die entgegengesetzte Situation wäre nicht akzeptabel. [10]

$$accuracy = \frac{correct\ predictions}{all\ predictions}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Abbildung 7: Formeln für Accuracy, Precision und Recall

Nachdem das Modell mit den Testdaten überprüft wurde, kann man beurteilen, ob die Genauigkeit die Erwartungen erfüllt. Falls dies nicht der Fall ist, bestehen drei Möglichkeiten die Genauigkeit zu verbessern.

Als Erstes kann man die ausgewählte Vorgehensweise für fehlende oder ausreißende Daten ändern oder anpassen und die zweite Möglichkeit ist es, den Algorithmus des Modells zu ändern. Am Ende kann man noch selbstständig die Hyperparameter anpassen, was jedoch ein mühseliger Prozess sein kann, um ihn zu verkürzen, kann man einen dieser Ansätze verwenden: Gridsuche, Zufallssuche, Bayessche Optimierung, Gradientenbasierte Optimierung oder Evolutionäre Optimierung. [8]

1.2.4 Optical Character Recognition

In den meisten Fällen erfolgt eine Eingabe über eine Tastatur, jedoch ist dies manchmal weder die beste noch effizienteste Art, Text einzulesen. Mithilfe von Optical Character Recognition (OCR) ist ein automatisiertes Einlesen und Verarbeiten von handschriftlichen oder gedruckten Text möglich und das schon bereits seit den 1950er. Am Anfang noch um Verkaufsberichte in Lochkarten zu konvertieren, damit ein Computer mit den Verkaufsdaten arbeiten kann. [11]

Im Bereich von OCR sind bereits gute und präzise Resultate mit Machine Learning (ML) erwartbar, jedoch wie bei allen anderen Problem ist es ausbaufähig. Um einen großen Fortschritt zu erreichen, würde die Nutzung von Deep Learning verpflichtend sein, dies ist jedoch in den meisten Situationen nicht notwendig.

Die Präzision hängt von vielen Attributen ab, daher kann ein eingescannter Text viel besser verarbeitet werden als ein im Freien geschossenes Bild mit dem Fokus auf ein Straßenzeichen [12].

- Textdichte

  Es macht einen Unterschied, wie viel Text sich auf einer Fläche oder einem Bild befindet, denn es ist in gewissen Situationen leichter, Text auszulesen, wenn dieser nur spärlich vorkommt.

- Struktur

  Wenn man eine klare Struktur erkennt, zum Beispiel in Tabellen oder in Zeilen, kann man ein besseres Ergebnis erwarten, daher ist es auch wichtig, dass man vor dem Auslese-Prozess das übergebene Bild aufbereitet und zum Beispiel die Rotation ändert.

- Schriftart

  Handgeschriebene Texte oder ”laute” Schriftarten sind im Gegensatz zu einfachen und gedruckten viel komplizierter, da sie kaum strukturiert sind. Außerdem könnten Buchstaben Ähnlichkeiten aufweisen, was später zu Verwechslungen führen kann.

- Buchstaben

  Sprachen wie Arabisch, Chinesisch, Russisch oder Japanisch benutzen im Gegensatz zu Deutsch ein anderes Alphabet, dabei kann es zu ähnlichen Buchstaben und Vertauschungen kommen, daher sinkt die Präzision in Texten mit mehreren Sprachen. Dies kann auch der Fall sein, wenn mathematische Formeln vorkommen.

Kyrillisches Alphabet	Ähnelt dem Buchstaben im Lateinischen Alphabet
р	p
В	B
Н	H
у	y
С	C

Tabelle 2: Ähnlichkeiten zwischen Buchstaben im Lateinischen und Kyrillischen Alphabet

- Platzierung

  Zentrierte Texte erlauben ein besseres Auslesen als abgeschnittene oder verstreute Wörter.

Die Texterkennung ist generell in zwei Phasen aufgeteilt:

- Text Detection
- Text Recognition

Text Detection

Text Detection ist der Prozess, indem in einem Bild oder einer PDF erkannt wird, wo sich Text befindet. Beim Resultat handelt es sich um Bounding Boxen, diese schließen einen Textblock (Wort, Buchstabe oder Paragraf) ein, dabei werden die genauen Koordinaten der Eckpunkte zurückgegeben. Diese werden später genutzt, um zur Visualisierung Boxen auf dem Bild oder der PDF aufzuzeichnen. Dieser Prozess wird auch in der Objekterkennung genutzt.

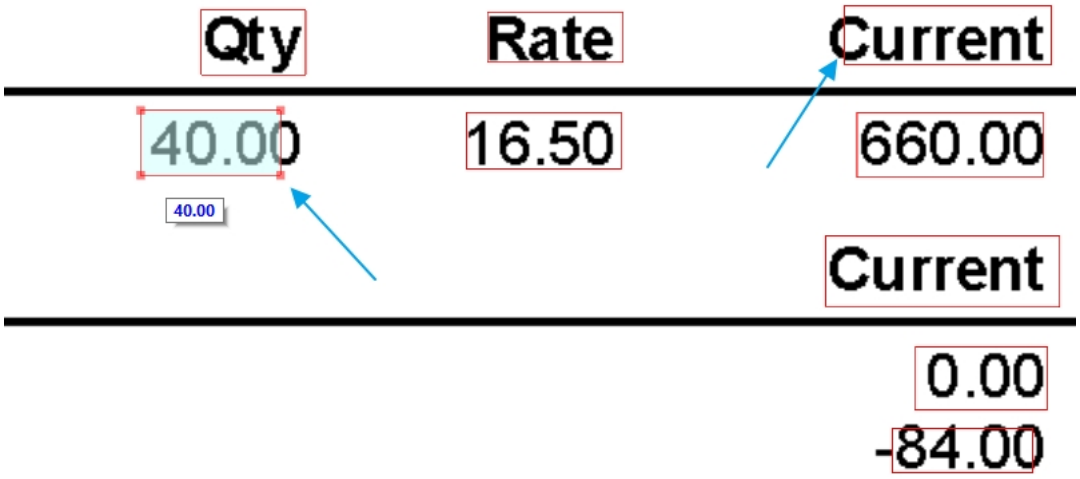

Abbildung 8: Bounding Boxes Beispiel

Die Erkennung kann entweder mittels der auf Regionen basierenden oder Textur basierenden Methode durchgeführt werden.

Bei der auf Regionen basierten Methode werden Pixel verbunden und als Zeichenkandidat markiert, welche später mehrmals gruppiert werden und schlussendlich Wörter oder Textzeilen bilden. Dabei kommt es auf die geometrischen Eigenschaften an, wo es zu Fehlern beim Gruppieren kommen könnte. Wie man bei Abbildung 8 sehen kann, wurde

der Buchstabe ”C” im Wort ”Custom” oder die letzte Nachkommastelle bei ”40.00” nicht ganz als Teil des Wortes oder der Zahl erkannt. [13]

Mit dem Stroke Width Transformer (SWT) wird jedem Pixel eine Strichbreite zugeteilt, indem zwei Kanten gefunden werden mit der gleichen Richtung modulo 180°. Die Entfernung dieser zwei Kanten werden in den Kantenpixeln und alle unterliegenden Pixeln als Strichbreite gespeichert und alle Zusammenliegeden gleich breite Pixel werden zu einem Zeichenkandidaten gruppiert. Danach werden alle benachbarten Zeichenkandidaten untersucht und zu einem Wort gruppiert, falls das mittelwertige Strichbreite-Verhältnis nicht über 2 liegt. Außerdem wird die Höhe und Farbe des Zeichenkandidaten berücksichtigt. Dies kann auch der Grund sein, wieso bei Abbildung 8 das Minuszeichen bei ”-84.00” nicht zur Zahl hinzugefügt wird, das Minuszeichen ist signifikant niedriger als der Rest der Zahl. [14]

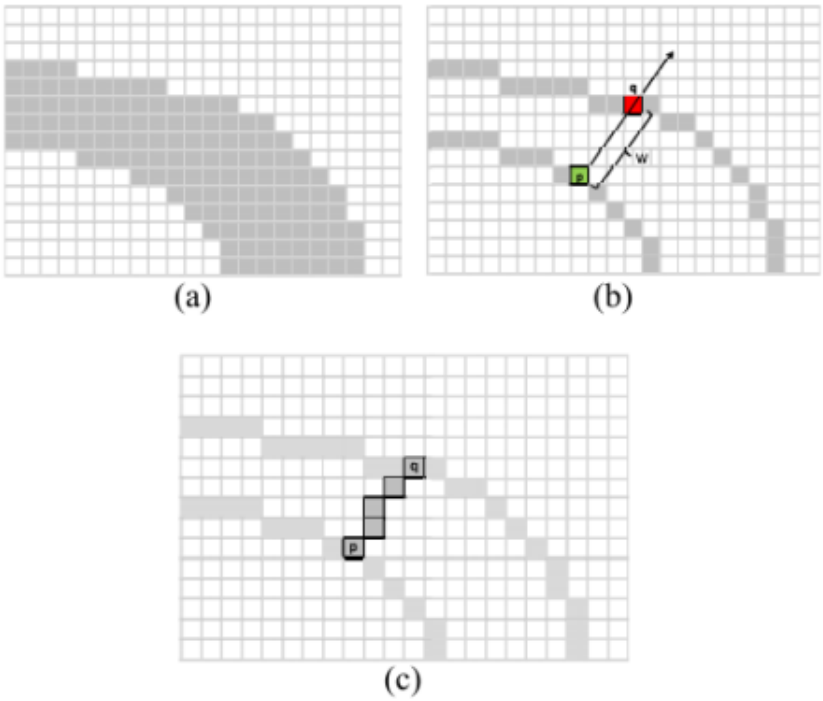

Abbildung 9: Die Kanten des Striches (a) werden solange verglichen, bis zwei gefunden werden mit der gleichen Richtung (b). Alle unterliegenden Pixel erhalten die Strichbreite der Entfernung zwischen der Start- und Endkante (c).

Textur basierte Methoden unterteilen das Bild in Fenster, dessen Höhe wird später mit der geschätzten Textgröße verglichen. Dabei kann es ebenfalls zu Erkennungsfehlern kommen. [13]

Die Mischung dieser beiden Methoden gewann im Jahr 2011 den ICDAR-Wettbewerb mit einem F-Score von 71.28%. Hierbei hat Chunghoon Kim den Vorschlag gegeben,

dass als erstes Blöcke mit dem Maximally Stable Extremal Regions (MSER) Verfahren extrahiert und danach benachbarte Blöcke gruppiert werden, falls die Farbe und Größe sich ähnelt. Jedoch werden mit diesem Verfahren ebenfalls eine große Menge an false-positive Blöcken erkannt. Um diese Anzahl zu verringern, wird eine ähnlich Idee zur SWT verwendet. [14]

Wettbewerbe sind ein großer Grund für die Fortschritte in der Texterkennung und im Rahmen des zwei-jährlich stattfindenden International Conference of Document Analysis and Recognition (ICDAR) Wettbewerbs wurden alle oben genannten Ideen verglichen [15].

Text Recognition

ist genau wie bei der Erkennung von Text in zwei Möglichkeiten unterteilt: Regionen basierend und Textur basierend.

Die von MSER generierten und normalisierten Blöcke werden je nach der Orientierung in ein separates Bild extrahiert. Dabei sind acht Orientierungen möglich, welche mit dem Gaußschen Filter bearbeitet werden und auf ein 5 x 5 Bild komprimiert wurden. Mit diesen 5 x 5 x 8 = 200 dimensionalen Vektoren werden dann die bereits erkannten Blöcke klassifiziert. [14]

1.2.5 Natural Language Processing

Natura Language Processing (NLP) gehört zu den Disziplinen der künstlichen Intelligenz, welche konstant überarbeitet und verbessert werden. Daher wird der Begriff NLP auch oft neuerfunden und anders interpretiert, dabei liegt der große Fokus auf das Wort "Processing" oder auf Deutsch "Verarbeitung". Das eigentliche Ziel von NLP liegt darin, eine für Menschen verständliche Sprache zu verstehen und damit umgehen zu können. Jedoch scheitert es daran, dass eine KI der heutigen Zeit nicht im Stande dazu ist, eine Schlussfolgerung aus einem Text zu ziehen. [16]

Die Texte reichen von täglichen E-Mails bis zu wissenschaftlichen Texten und sich aus dem Alltag nicht mehr wegdenkbar. Zu diesen Aufgabenbereichen gehören zum Beispiel: [17]

- Spam Erkennung

Ohne NLP wären alle E-Mail-Postfächer voll mit Spammails. Dabei schaut ein NLP-Modell auf die Grammatik, gewisse Begriffe und falsch geschriebene Wörter, da Spam-Emails oft die gleichen Charakteristiken von frei übersetzten Texten aufweisen.

- Maschinen Übersetzungen

  Da der Sinn eines Satzes nicht verloren gehen soll, wird beim Übersetzen NLP genutzt, um den Kontext zu übernehmen.

Darüber hinaus wird NLP in verschiedene Bereiche unterteilt.

1.2.6 Named Entity Recognition

Unter Named Entity Recognition (NER) versteht man das Extrahieren und Identifizieren von Entitäten in einem Text. Wobei es sich bei dem Typen um eine Zahl, ein Wort oder eine Zusammensetzung von mehreren Wörtern handeln kann.

Dieser Vorgang ist in zwei Schritte unterteilt: [18]

- Identifizierung einer Entität
- Klassifizierung einer Entität

Die Identifizierung versucht in einem Text vorkommende Wörter in Entitäten und Füllwörter zu unterteilen. Dazu ist es wichtig, dass ein Modell nicht nur mit Entitäten antrainiert wird, sondern mit einem ganzen Text, der beide Wortarten beinhaltet.

Die Klassifizierung versucht einer Entität eine Klasse zuzuweisen. Die meisten NER-Modelle sind schon mit herkömmlichen Klassen (Person, Ort, Uhrzeit, Datum) antrainiert, jedoch besteht auch die Möglichkeit, eigene Klassen zu erstellen und das Modell mit diesen zu trainieren.

1.3 Deep Learning

Sowie beim ML sind DL-Algorithmen abhängig von antrainierten Daten und kann daher als Synonym oder Untergruppe gesehen werden. Der grobe Unterschied liegt darin, dass diese Daten meist komplett roh sind und keine Vorarbeit geleistet wurde. Daher kann ein DL-Modell wie ein Mensch noch nie davor gesehene, spezielle Bilder kategorisieren.

Die Einsatzmöglichkeiten überlappen sich sehr mit jenen von ML, dennoch ist das Resultat von DL-Modellen viel präziser. Daher würden die meisten modernen Assistenten ohne DL auch nicht auf dem erwarteten Niveau arbeiten.

Obwohl schon viele Anwendungen auf DL basieren, handelt es sich dabei um eine junge Technologie, die noch weit von ihrem vollen Potenzial entfernt ist.

1.4 ML vs DL

1.4.1 Feature Extraction

In beiden Fällen unterlaufen die Daten denselben Phasen: Feature Extraction und Classification. Jedoch unterscheiden sie sich bei der Ausführung der Feature Extraction.

In der Feature Extraction werden Merkmale zusammengefasst und in einer strukturierten Tabelle dargestellt. Jede Zeile beinhaltet diese Merkmale und die dazugehörige Klassifizierung.

Bei einem ML-Modell ist die Feature Extraction ein separater Vorgang, welcher nicht automatisch durch das Modell vorgenommen wird. Dieser Prozess ist oft sehr kompliziert und beansprucht eine lange Zeit, außerdem ist hier eine Person notwendig, die sich in dem gegebenen Bereich auskennt.

Im Kontrast dazu nutzt ein DL-Modell ein Neuronales Netzwerk, welches sich stark an einem menschlichen Gehirn orientiert und die Feature Extraction übernimmt.

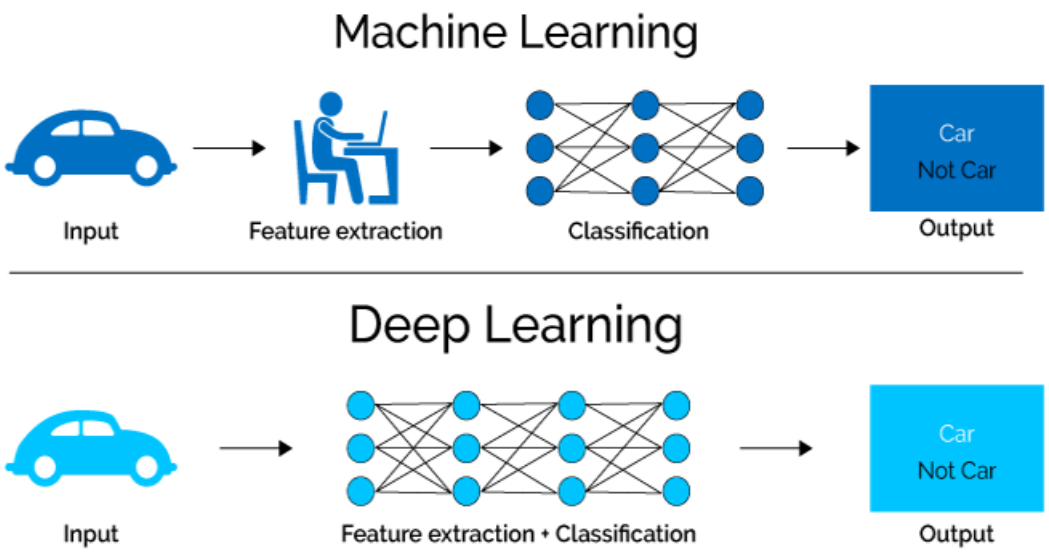

Abbildung 10: Feature Extraction, ML vs DL

Wie der Name ”Neuronales Netzwerk” schon verrät, besteht dieses Konzept aus Tausenden nachgeahmten Neuronen, die über mehrere Schichten zusammen kommunizieren. Dabei kann sich ein Neuron nur mit anderen Neuronen auf derselben oder darüberlegenden Schicht verknüpfen, daher entsteht eine hierarchische Struktur.

Die einzelnen Schichten können mit den fünf Sinnen eines Menschen verglichen werden. Verbindet man zum Beispiel einer Person die Augen und lässt sie dann eine Erdbeere probieren, kann die Person anhand des Geschmackes und des Duftes erkennen, um welche Frucht es sich handelt. Ähnlich würde es in einem Neuronalen Netzwerk ablaufen, wobei der Geschmack und der Duft einzelne Schichten repräsentieren.

Legt die Person daraufhin die Augenbinde ab und sieht, dass die Erdbeere weiß ist, kann sie sich trotzdem sicher sein, dass es sich um eine Erdbeere handelt, da sie aus Erfahrung sagen kann, dass der Geschmack sowie der Duft eindeutiger sind. Dies würde ein Neuronales Netzwerk am Anfang verwirren, da es nicht über diese Erfahrung verfügt. Mit der Zeit würde das Neuronale Netzwerk bestimmten Synapsen (Verbindungen) eine Gewichtung zuteilen, welche die Wichtigkeit einzelner Parameter angibt. Dieses Einordnen ist die Art, wie ein Neuronales Netzwerk ”lernt”. [19]

1.4.2 Vergleich

Ein ML-Modell sollte bevorzugt in Bereichen genutzt werden, wo bereits strukturierte Daten vorhanden sind. Die Menge an Daten ist hierbei nicht sehr relevant, da ein ML-Modell mit wenigen Daten präzise sein kann.

Stehen keine bereits präparierten Daten zur Verfügung, wäre ein DL-Modell die bessere Lösung, da man sich die Zeit und Kosten für die Feature Extraction sparen kann. Jedoch benötigt man leistungsstarke Computer, da Neuronale Netzwerke komplexe Rechnungen durchführen, die mithilfe einer GPU beschleunigt werden können. Außerdem ist die Programmierung und Trainingsphase sehr lang und mühselig. [20]

Bei der Entscheidung, welches Modell sinnvoller ist, sollte man außerdem beachten, dass die Technologie noch nicht voll entwickelt ist und dass manche Probleme mit dem momentanen Stand der Technik nicht lösbar sind.

2 Ausgangslage

2.1 Ausgangssituation

Als Mitglied der weltweiten Bechtle Gruppe stellt smartpoint IT consulting kundenspezifische Softwarelösungen im Bereich Office365, SharePoint und Dynamics 365/CRM bereit. Mit Standorten in Linz, Wien und Graz beschäftigt das Unternehmen über 100 Mitarbeiter, die mit laufenden Weiterbildungen immer auf dem neuesten Stand der Technologien bleiben und somit für das optimale Ergebnis sorgen.

Zusammen mit zahlreich namhaften Unternehmen aus vielen unterschiedlichen Branchen werden seit 2007 Projekte gestartet, die noch bis heute laufen. Dabei bieten sie in allen Phasen der Projektentwicklung Unterstützung an, mit einer Spezialisierung auf Cloudlösungen. Dazu wird die User Experience und das Design auf keinster Weise vernachlässigt.

Unter anderem bietet smartpoint folgende Dienstleistungen an:

- Beratung
- Konzeption
- Umsetzung
- Betreuung der Systeme

Obwohl die Welt um den Bereich Künstliche Intelligenz immer größer und wichtiger wird, wird sie nur wenig in smartpoints Projekte eingebaut und daher sind auch nur wenig bis keine Erfahrungen im Bereich Künstliche Intelligenz vorhanden.

2.2 Istzustand

Momentan wird bei der Konzeptionierung von Projekten mehr auf logische Abfolgen Wert gelegt anstatt auf die Nutzung von Künstlichen Intelligenzen. Wird jedoch eine Künstliche Intelligenz benötigt, greift man auf die bereits vorimplementierten Produkte von Microsoft wie Power Automate und dem AI Chatbot zurück.

2.3 Ziele

Um in Zukunft das Potential von Cloud-basierten Softwarelösungen in Microsoft Azure in Kombination mit Künstlicher Intelligenz besser ausschöpfen zu können, soll im Rahmen der Diplomarbeit eine Gegenüberstellung von verschiedenen Möglichkeiten ausgearbeitet werden. Dabei sollen sowohl Low-Code-Lösungen (AI Builder, Cognitive Services) als auch eigene Implementierungen mit Python als Azure Functions sowie deren Anbindung an andere Systeme (z.B. Dynamics 365) getestet werden.

Aufgrund von organisatorischen Differenzen wurde die Diplomarbeit in zwei Teile geteilt, wobei die Implementierungen mit Python als Azure Function in diesem Teil behandelt wird.

2.4 Aufgabenstellung

Da es sich um eine theoretische Situation handelt, wurde zusammen mit dem Projektpartner ein für das Unternehmen relevantes Szenario ausgearbeitet. Dabei handelt es sich um das Verarbeiten von externen Rechnungen, wobei folgende Felder ausgelesen werden müssen:

Unternehmens Name	Der Name des Unternehmens, das die Rechnung ausstellt
Unternehmens Adresse	Die Adresse des Unternehmens, das die Rechnung ausstellt
Rechnungs Id	Eine eindeutige Nummer, die einer Rechnung zugewiesen ist
Rechnungs Datum	Das Datum, an dem die Rechnung ausgestellt wurde
Gesamtbetrag	Der Gesamtbetrag der Rechnung
Rechnungszeilen	Jedes Produkt oder jede Dienstleistung, die in der Rechnung aufgelistet wurde (inklusive Beschreibung, Anzahl, Stückpreis und Gesamtpreis)

Tabelle 3: Geforderten Felder beim Auslesen einer Rechnung

3 Umsetzung

3.1 Verwendete Technologien

3.1.1 Frontend

HTML

Mit der steigenden Nachfrage von Webseiten in den 1990er wurde Hypertext Markup Language (HTML) als strukturierte Möglichkeit vorgestellt, um grafische Webseiten zu erstellen.

Die Basis jeder Webseite ist ein HTML-Gerüst, welches die Struktur und Einzelteile angibt, dabei umfasst der HTML-Standard Überschriften, Paragrafen, Tabellen und Listen. Zusätzlich ermöglicht HTML das Inkludieren von Fotos, Videos als auch von anderen Dokumenten (PDF-Dateien, Webseiten). [21]

Um das Arbeiten mit HTML zu vereinfachen, wurden genau wie bei anderen Technologien für jedes Programm unterschiedliche Plugins eingebunden. Mit der Zeit wurden die Plugins in den Standard eingefasst und damit entwickelten sich über die Jahre unterschiedliche Versionen. Seit Oktober 2014 ist HTML5 als allgemeiner Grundsatz definiert und erhielt seit jeher keine weitere Versionsnummer, da man Änderungen als ”living standard” weiterentwickelt. [22]

Ein HTML-File (.html, .htm) besteht aus HTML-Tags, welche mit einem Kleiner-Zeichen ”<” beginnen und einem Größer-Zeichen ”>” enden. Der Inhalt dazwischen unterscheidet sich je nach Komponente (p, h1, a, ...).

Neben dem eigentlichen Tag können HTML-Elemente noch zusätzlich Attribute beinhalten, welche zusätzlich Informationen liefern. Diese werden normalerweise als Name/Value-Paar im Starting-Tag aufgelistet. Dabei sind die meisten freiwillig, jedoch gibt es dafür auch Ausnahmen (src-Attribut im img-Tag). [23, 24]

In der Regel besteht jede Komponente aus zwei HTML-Tags, einem Starting-Tag und einem Ending-Tag, welches noch einen zusätzlichen Schrägstrich ”/” beinhaltet. Jedoch gibt es Self-Closing-Tags als Ausnahme, wo eine Komponente aus einem einzelnen

HTML-Tag besteht. Dazu gehört das `<img>`-Tag, da es nur aus einem Starting-Tag und Attributen besteht.

Eine HTML-Datei nimmt die Struktur eines Baumes an, das heißt, dass es sich um eine hierarchische Abfolge von Komponenten handelt. Diese Struktur ähnelt der von Extensible Markup Language (XML), jedoch befolgt es nicht alle Richtlinien von XML. Wird diese Kompatibilität vorausgesetzt, muss man auf Extensible Hypertext Markup Language (XHTML) zurückgreifen. [21]

Zu den wichtigsten HTML-Tags gehören: [25]

`<h1>` bis `<h6>`	Überschrift in unterschiedlicher Reihung (1 = Hauptüberschrift, 2-6 = Unterüberschriften)
`<p>`	Paragraph
`<a>`	Hyperlink
`<img>`	Bild
`<ol/ul>`	Geordnete und ungeordnete Liste
`<li>`	Element einer Liste
`<strong>`	Hebt den Text in Fettschrift hervor
`<div>`	Block-Element
`<table>`	Tabelle
`<tr>`	Tabellenzeile

Tabelle 4: Beispiele für HTML-Tags

Zusätzlich könnten HTML-Elemente diese Attribute enthalten (Auszug):

`id`	Mit der Id kann ein HTML-Element direkt referenziert werden
`title`	Gibt Zusatzinformationen zu einem HTML-Element, welche als Tooltip dargestellt werden
`disabled`	Gibt an, ob ein HTML-Element benutzbar ist (benötigt keinen Wert)

Tabelle 5: Beispiele für HTML-Attribute

Ein gewöhnliches HTML-File beginnt mit `<!DOCTYPE html>`, um den Browser mitzuteilen, welcher Datentyp zu erwarten ist. Diese Information unterscheidet sich je nach Version und bei XHTML. [26]

Ein HTML-File, welches einen Auszug aus den wichtigsten HTML-Tags beinhaltet, könnte so aussehen:

Listing 2: Beispiel für ein HTML-File

```
<!DOCTYPE html>
<html>

<head>
    <title>Titel - DA</title>
</head>

<body>
    <!-- Dies ist ein Zeilenkommentar, der sich ueber mehrere Zeilen ziehen kann.
        -->
    <h1>Ueberschrift 1</h1>
    <p>Ein Paragraph, der mit Abstand unter der Ueberschrift angezeigt wird.</p>
    <ul>
        <li>Elemente einer</li>
        <li>ungeordneten Liste.</li>
    </ul>
    <p>
        Ein Paragraph mit
        <br /> <!-- Dies ist ein Beispiel fuer ein Self-Closing Tag -->
        einer Unterbrechung.
    </p>
</body>

</html>
```

Wird das vorherige HTML-File in einem gewöhnlichen Browser geöffnet, wird die Struktur folgendermaßen interpretiert:

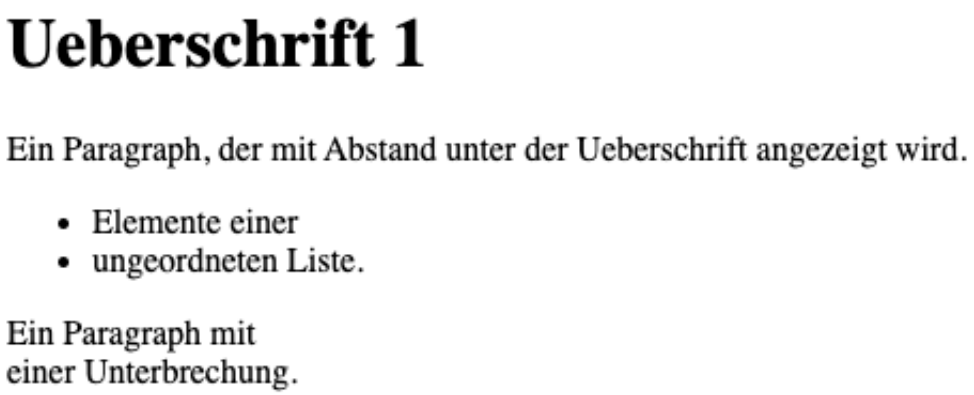

Abbildung 11: Beispiel für ein angezeigtes HTML-File

CSS

Um Webseiten zu gestalten, wird Cascading Style Sheets (CSS) verwendet, damit kann das Design von der generellen Struktur und dem Inhalt getrennt werden. Wobei CSS immer mit HTML assoziiert wird, kann es mit jeder auf XML basierenden Ausdruckssprache benutzt werden. [21]

Generell besteht eine CSS-Regel aus einem Key/Value-Pair, welches die Styling-Property (was soll geändert werden) und den Styling-Wert (auf was soll es geändert werden) setzt. Dabei wird der Styling-Wert je nach Styling-Property eingeschränkt.

Zusätzlich können mehrere CSS-Regeln zu einer zusammengefasst werden, diese Regeln werden als Shorthand bezeichnet (z.B.: `background`, `border`). Abhängig von der Werteanzahl wird ein bestimmtes Shorthand referenziert. Die Styling-Property `margin` kann folgendermaßen angegeben werden:

- Ein Wert: `margin: 10px`

  Dieser Wert wird für alle Seiten genutzt.

- Zwei Werte: `margin: 10px 5px`

  Der erste Wert wird für die obere und untere Seite genutzt und der zweite für die linke und rechte.

- Drei Werte: `margin: 10px 5px 15px`

  Der erste Wert wird für die obere Seite genutzt, der zweite Wert für die linke und rechte Seite und der letzte Werte für die untere.

- Vier Werte: `margin: 10px 5px 15px 20px`

  Die Werte werden im Uhrzeigersinn beginnend mit der oberen Seite vergeben.

Beispiele für gültige Properties:

`width/height`	`100px` / `100%`	Gibt die Breite/Höhe eines HTML-Elements an (als Länge oder Prozentzahl)
`display`	`block` / `flex`	Gibt an, wie die HTML-Elemente angezeigt werden
`margin/padding`	`10px`	Gibt den inneren/äußeren Abstand an
`background`	`#fff`	Gibt die Farbe/das Bild für den Hintergrund an
`color`	`red`	Gibt die Farbe/Strichstärke an

Tabelle 6: Beispiele für CSS-Properties

Um die Struktur mit dem Styling zu verbinden, bietet HTML drei Möglichkeiten an: [27]

- Direkt im HTML-File innerhalb vom `<head>`-Tag mithilfe von einem `<style>`-Tag.

```
<head>
    <style>
        <!-- CSS-Regeln -->
    </style>
</head>
```

- Als Referenz zu einem separatem CSS-File (Dateinamenserweiterung: `.css`) wird ein `<link>`-Tag im `<head>`-Tag gesetzt. Wobei diese Möglichkeit es ermöglicht, die CSS-Regeln in mehreren HTML-Files zu nutzen und man damit Codeverdoppelung vermeidet.

```
<head>
    <link rel="stylesheet" href="pfad/zum/css-file.css">
</head>
```

- Direkt im HTML-Tag mit dem `style` Attribut.

```
<div style="<!-- CSS-Regeln -->">
```

Um eine CSS-Regel zuweisen zu können, werden sogenannte Selektoren angegeben, welche direkt ein HTML-Element ansprechen oder mehrere zusammenfassen. Gültige Selektoren sind:

`*`	`*`	Selektiert alle HTML-Elemente
`element`	`h1`	Selektiert alle HTML-Elemente mit dem angegebenen Elementnamen
`.klasse`	`.nav`	Selektiert alle HTML-Elemente mit dem angegebenen `class`-Attribut
`#id`	`#main`	Selektiert alle HTML-Elemente mit dem angegebenen `id`-Attribut

Tabelle 7: CSS-Selektoren

Um diese Spezifikation noch genauer zu definieren, können Selektoren mit unterschiedlichen Operatoren zusammengefügt werden.

`,`	`h1, p`	Selektiert alle `<p>` und `<h1>`-Elemente
” ” (Leerzeichen)	`div p`	Selektiert alle `<p>`-Elemente in einem `<div>`-Element
`>`	`div > p`	Selektiert alle `<p>`-Elemente, die direkt in einem `<div>`-Element sind
(2 Selektoren ohne Operator)	`div.main`	Selektiert alle `<div>`-Element, die auch die Klasse `.main` haben

Tabelle 8: CSS-Operatoren

SCSS Da die Funktionalitäten von CSS sehr eingeschränkt sind, wird oftmals als Weiterentwicklung Sassy Cascading Style Sheets (SCSS) genutzt. Dies ermöglicht das Vereinfachen von Regeln und macht diese generell überschaubarer.

Zusätzlich zu den normalen Funktionen von CSS stellt SCSS zum Beispiel folgende zur Verfügung: [28]

- Variablen

  Um über mehrere Regeln einen gleichen Wert zu setzen, kann dieser Wert in einer Variable gespeichert werden. Dies ermöglicht ein leichtes Updaten an mehreren Stellen gleichzeitig.

- Nesting

  Da mit der Zeit die Selektoren sehr kompliziert werden, ermöglicht SCSS das Erstellen von Regeln in einer hierarchischen Abfolge.

- Modules

  Damit man Codeverdoppelung vermeidet, kann man weitere CSS-Files importieren.

Angular

Angular ist ein auf TypeScript basierendes Frontend-Framework, welches neben React, Express und Vue.js das beliebteste Framework auf dem Markt ist. [29] Dabei wird Angular sowohl für kleine Projekte in der Schule als auch große Projekte in weltbekannten Plattformen verwendet.

Components Das Frontend wird in Komponenten (Components) aufgeteilt und diese stellen unterschiedliche Elemente dar, welche auch mehrmals untereinander verwendet werden können, um Codeverdopplung zu verhindern.

Jedes Component besteht aus

- einer Component-Klasse (TypeScript-File),
- einem HTML-Template und
- einem Stylesheet (CSS/SCSS-File).

Listing 3: Beispiel eines Angular-Components

```
@Component({
    selector: 'custom-app',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.scss']
})
export class AppComponent {

title: string = "Meine eigene App!";

items: any[] = [
    {
        "text": "Das ist meine"
    },
    {
        "text": "erste eigene Angular-App."
    }
];

// Restlicher Code
}
```

Wie man im obigen Beispiel sieht, müssen bei einem Component Felder gesetzt werden, damit die unterschiedlichen Files miteinander assoziierbar sind:

selector	verpflichtend	Gibt den HTML-Tag an, mit welchem man das Component in anderen aufrufen kann (Bsp.: <app-custom-app></app-custom-app>)
template/templateUrl	verpflichtend	Gibt entweder direkt den HTML-Code (template) an oder referenziert zu einem HTML-File (templateUrl)
styles/styleUrls	freiwillig	Gibt entweder direkt einen oder mehrere CSS-Codes (styles) an oder referenziert zu einem oder mehreren CSS-Files (styleUrls)

Tabelle 9: Parameter einer Komponente

HTML-Templates Neben der generellen Struktur des Frontends kann ein HTML-Template auch Daten aus dem Component anzeigen.

Listing 4: Beispiel fuer ein HTML-Template

```
<h1>{{title}}</h1>

<ul>
    <li *ngFor="let item of items">{{item.text}}</li>
</ul>
```

Generell besteht die Struktur aus üblichen HTML-Tags, jedoch kann es zusätzlich noch Component-Tags oder auch programmierähnliche Operationen beinhalten. Hierbei unterscheidet man zwischen:

- Interpolation

- Property Binding
- Event Binding
- Two-Way Binding

Interpolation Bei einer Interpolation werden zwei geschwungene Klammern "{{}}" genutzt, um den Wert einer Variable im Frontend anzuzeigen. Dabei muss darauf geachtet werden, dass die Interpolation immer den String-Wert der Variable nutzt. [30]

Property Binding Mit einem Property Binding können genau wie bei der Interpolation Daten im Frontend angegeben werden, jedoch kann jeder Datentyp genutzt werden.

Event Binding Damit kann das Backend auf gewisse Aktionen (Klicken, Bewegung der Maus, Hover) im Frontend reagieren.

Two-Way Binding Die Mischung aus den Vorherigen ergibt das Two-Way Binding. Sowohl das Backend als auch das Frontend reagieren auf Änderungen der jeweils anderen Komponente.

CLI Ein Command Line Interface (CLI) ermöglicht das Nutzen von Funktion über die Kommandozeile, spezifisch für Angular wird darüber das Erstellen, Generieren und Verwalten gesteuert. Das Kommandowort für Angular lautet `ng` und darauf können unterschiedliche Funktionswörter folgen.

TypeScript

Laut der jährlichen Entwicklerumfrage von StackOverflow landet TypeScript nach Python auf dem zweiten Platz (mit 15,29%) der meistgesuchten Programmiersprachen. Damit liegt TypeScript mit 0,7% vor JavaScript, die Sprache, auf der TypeScript basiert. [29]

TypeScript versucht JavaScript zu verbessern, indem es eine strikte Typisierung fordert. Damit beugt man dem einfachen Fehler vor, dass Werte einen anderen Typ als für sie vorgesehen enthalten, als der gefordert ist. Dies führt dazu, dass Typenfehler bereits

in der Entwicklung gefunden werden und verhindert, dass diese Fehler überhaupt in der Produktivumgebung aufkommen. Abgesehen davon sind die zwei Sprachen ident, daher ist es auch möglich, bereits bestehende Tools von JavaScript mit TypeScript zu verwenden. Das heißt, dass alle JavaScript-Libraries weiterhin genutzt werden können. Die beliebtesten Libraries sind zum Beispiel jQuery und Chart.js. [31]

3.1.2 Backend

Python als Progrmmiersprache

Nach dem Scheitern seiner ersten Programmiersprache entwickelte Guido van Rossum die Sprache Python, dabei wollte er alle Fehler, die er beim Entwickeln von ABC gefunden hat, verbessern. Daher basieren auch die Strukturen und Konventionen von Python auf Unix, ohne an Unix gebunden zu sein. [32]

Python unterscheidet sich in vielen Punkten zu anderen Sprachen, unter anderem, dass sie viel Wert auf Lesbarkeit legt. Die auffälligste davon ist, dass Einrückungen Codeblöcke unterteilen anstatt eine Art von Klammern. Dafür gibt es zwei Gründe:

- Es macht den Code kürzer und er wirkt nicht unnötig ausgeschmückt, daher braucht man eine kürzere Aufmerksamkeitsspanne, um den Sinn einer Codestelle nachvollziehen zu können.
- Die Struktur des Codes ist vereinheitlicht, was es einfacher macht, Projekte von anderen zu verstehen.

Außerdem werden dem Entwickler in vielen Entscheidungen leichter gemacht, da unnötige Möglichkeiten entfernt wurden, das heißt, dass es meistens nur eine offensichtliche Art und Weise gibt, etwas zu implementieren. Dazu kommt noch die Nutzung von Spezialzeichen. Es werden nur Zeichen unterstützt, die den meisten bereits bekannt sind und dessen Operation Sinn machen. [32]

Dies beantwortet jedoch nicht die Frage ”Wieso ist Python die beliebteste Sprache für ML?”. Die Antwort darauf ist, dass nicht nur triviale Aufgaben bereits vorimplementiert, sondern auch, dass die meisten ML Funktionen in Python Libraries zusammengefasst sind. Daher muss man als Neuanfänger oder Fortgeschrittener keine bereits gelösten Probleme von Grund auf noch einmal angehen.

Notebooks

Da es bei ML öfters dazu kommt, dass bestimmte Codeblöcke oft wiederholt ausgeführt werden, kann man mithilfe von Notebooks diese einzeln ausführen. Zum Beispiel beim Analysieren eines Datensatzes oder um schnell kleine Änderung am geplanten Vorgehen vorzunehmen.

Diese Codeblöcke können entweder Code, Texte oder Grafiken beinhalten, diese werden fortlaufend mit einer Nummer versehen und die Ausgabe/Grafik erscheint direkt unter dem Code.

```
[1]: from matplotlib import pyplot as plt
     import numpy as np

     # Generate 100 random data points along 3 dimensions
     x, y, scale = np.random.randn(3, 100)
     fig, ax = plt.subplots()

     # Map each onto a scatterplot we'll create with Matplotlib
     ax.scatter(x=x, y=y, c=scale, s=np.abs(scale)*500)
     ax.set(title="Some random data, created with JupyterLab!")
     plt.show()
```

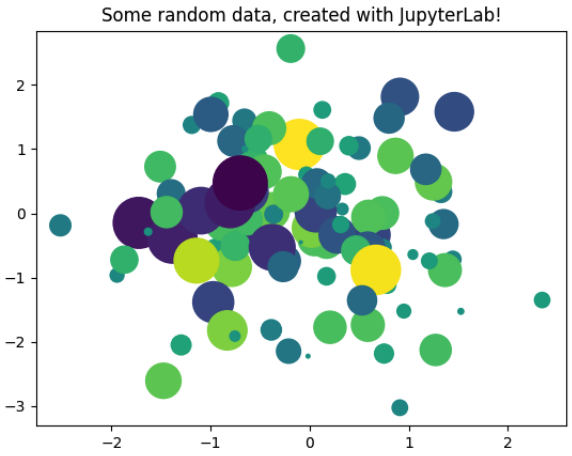

Abbildung 12: Beispiel für ein Notebook mit Code und Ausgabe mit Jupyter Notebook

Diese Notebooks kann man entweder lokal starten oder auf diversen Webseiten online benutzen, dann wird auch der Code auf den Servern dieser Notebookseiten ausgeführt. Dies kann von großem Vorteil sein, da oft generierte Grafiken sehr viel Leistung in Anspruch nehmen. Außerdem werden die Ergebnisse/Ausgaben gespeichert und man muss nicht bei jedem Neuladen jeden Codeblock neu ausführen.

Libraries

Pandas Bevor der eigentliche ML-Prozess beginnen kann, müssen die Daten als Erstes analysiert werden und dafür ist die Library ”pandas” geeignet. Neben der Analyse ist pandas auch in der Datenmanipulation sehr hilfreich, da die vordefinierten Funktionen

nicht nur leicht und verständlich, sondern auch für echte Daten aus der Welt gedacht, sind.

Zu den zwei Hauptdatentypen von pandas gehören Dataframes und Series.

Series sind eindimensionale Arrays, welche aus verschiedenen Datentypen bestehen können. Mithilfe von einer ID kann man auf jeden Eintrag in der Series zugreifen, diese sind entweder mitgegeben worden oder von pandas generiert (aufsteigende Zahl beginnend bei 0).

Mit der Methode `pandas.Series(data)` können Dictionaries, ndarrys und Skalarwerte in Series umgewandelt werden.

Dataframes sind zweidimensionale Tabellen mit verschiedenen Datentypen als Spalten. Sie sind vergleichbar mit Datenbanktabellen oder mit einem Dictionary mit dem Datentyp Series als Value.

Jede Zeile beinhaltet eine ID, falls diese nicht vergeben ist, wird von pandas eine generiert und mittels dieser ID kann man dann auf bestimmte Zeilen zugreifen. Das gleiche Prinzip gilt bei den Spalten, wo man mit der Spaltenbezeichnung auf jeden Wert der Zeilen assoziiert und mit dieser Spalte zugreifen kann.

pandas bietet beim Erstellen von Dataframes mehrere Möglichkeiten an: [33]

- Einlesen einer Datei (*.json, *.html, *.sql, *.xlsx, ...)

  Am Beispiel einer .csv-Datei: `pandas.read_csv(FILENAME)`

- Dictionaries

  `pandas.DataFrame.from_dict(DICT_VARIABLE)`

- Series

  `pandas.Series.to_frame(SERIES_VARIABLE)`

- Anderes Dataframe

Das Benutzen von pandas erspart sehr viel Zeit und ermöglicht den schnellsten Weg zu einem Ergebnisse mit diesen Funktionalitäten:

- Löschen und Ersetzen von fehlenden Daten, was beim Vorbereiten der Daten hilfreich ist

- Hinzufügen und Löschen von Spalten in Dataframes
- Gruppierung von Daten
- Koventierung in NumPy oder andere Python Objekte
- ...

spaCy spaCy ist eine Library, die beim Implementieren von NLP-Modell vieles vereinfacht. Mit der Veröffentlichung in 2015 überholte es schnell die zu der Zeit meistgenutzte NLP-Library NLTK. Als öffentliche Library wurde sie immer vermehrt in privaten als auch kommerziellen Projekten verwendet. [34]

Mit dem Download von spaCy werden insgesamt 64 Sprachen mitgeliefert, welche bereits mit einem NLP-Modell referenziert sind. Diese Modelle wurden mit einer großen Menge an Daten für unterschiedliche Bereiche trainiert. Reichen diese jedoch nicht, gibt es immer die Möglichkeit, ein bereits Existierendes anzupassen oder ein komplett Neues zu erstellen. [34]

spaCy stellt sogar ein antrainiertes NER-Modell zur Verfügung, welches Entitäten erkennt, die im kommerziellen Bereich oft zum Beispiel in Verträgen, Artikeln oder Rechnungen vorkommen. Die in Tabelle 10 aufgezählten Entitäten, werden mithilfe von einem vortrainierten NER-Modell erkannt.

Das Endergebnis kann aus mehreren Zeilen bestehen und setzt sich aus vier Teilen zusammen. Zuerst der eigentliche Text, danach der Startindex und Endindex im Text und zum Schluss die erkannte Entität.

Listing 5: Beispiel für ein spaCy NER Ergebnis

```
1     [
2         ('fb', 0, 1, 'ORG'),
3         ('Barack Obama', 13, 25, 'PERSON'),
4         ('Wels', 94, 98, 'GPE'),
5         ('1994', 0, 4, 'DATE')
6     ]
```

Zu diesen zuvor antrainierten Entitäten gehören: [34]

PERSON	Person (Name)
NORP	Nationalitäten oder religiöse oder politische Gruppe
FAC	Gebäude, Flughafen, Brücken, ...
ORG	Unternehmen, Agentur, Institution, ...
GPE	Länder, Städte, Bundesstaaten
LOC	Nicht GPE Orte (Meere, Seen, Berggruppen)
PRODUCT	Objekte, Fahrzeuge, Essen, ...
EVENT	Kriege, Kämpfe, Sport Events, ...
WORK_OF_ART	Büchertitel, Songs
LAW	Gesetze
LANGUAGE	Sprachen
DATE	Datum
TIME	Uhrzeiten, die kleiner als ein Tag sind
PERCENT	Prozentzahl
MONEY	Geldanzahl (mit Währung)
QUANTITY	Messungen von Gewichten oder Längen
ORDINAL	Aufzählungen
CARDINAL	Zahlen, die zu keinem anderen Typen passen

Tabelle 10: spaCy NER Entitäten

3.2 Frontend

Das auf Angular basierende Frontend spiegelt die unterschiedlichen Ansätze wieder und stellt zusätzlich Tools zur Verfügung.

3.2.1 NER-Ansatz

Der NER-Ansatz basiert auf einem selbst antrainierten und programmierten Modell und ermöglicht sowohl das Aufbereiten der Daten als auch das Extrahieren der Daten aus einer Rechnung. Daher werden diese zwei Schritte unterteilt in:

- NER Annotation Tool
- Invoice Reader

NER Annotation Tool

Bei dem NER Annotation Tool handelt es sich um ein selbst programmiertes Tool, welches es einem erleichtert, Rechnungen oder übliche Texte im pdf-Format zu klassifizieren.

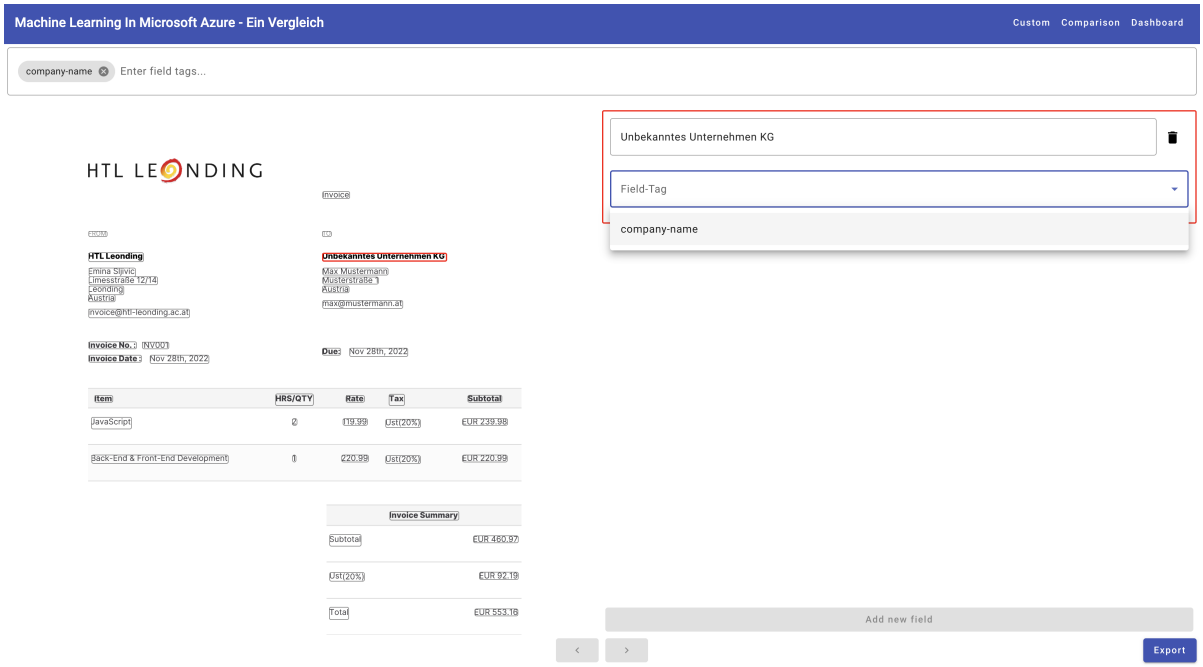

Abbildung 13: NER Annotation Tool mit Beispieldaten

Nachdem der User eine oder mehrere Dateien ausgewählt hat, wird im Hintergrund mithilfe von Event-Binding eine Methode ausgelöst, die entweder einen oder mehrere Hypertext Transfer Protocol (HTTP)-GET-Requests abschickt. Diese werden als Promise gespeichert, was heißt, dass das Programm auf alle Antworten wartet, bevor es mit dem nächsten Schritt weiter macht. Die Antworten geben die ausgewählten pdf-Dateien als blob-Objekt zurück.

Listing 6: HTTP-GET-Requests für die pdf-Dateien

```
1       const getImagePromises = this.invoices.map(i => this.http.get(i.url,
            {responseType: 'blob'}).toPromise());
2       const images = await Promise.all(getImagePromises);
```

Dieses blob-Objekt wird durch das zusätzlich installierte Paket `ng2-pdf-viewer` dargestellt.

Mit weiteren HTTP-Requests werden die Azure Functions aufgerufen, um das pdf-File richtig zu skalieren und um jede Wortgruppe mit einer Box zu umranden. Mithilfe dieser Boxen kann die Rechnung annotiert werden. Jede Annotation besteht aus dem Inhalt der Box und einer Entität, die zuvor im oberen Feld eingegeben wurden.

Um diese Boxen darzustellen, wird automatisch ein `div`-Element über dem pdf-Viewer erstellt und durch den `renderer` mit den Boxen gefüllt. Ein `renderer` ermöglicht das Eingreifen in den HTML-Struktur über den dahinterliegenden Code.

All diese HTTP-Requests werden mit dem über Dependency-Injection initialisierten HttpClient ausgeführt. Dieser Client wird von Angular zur Verfügung gestellt.

Nachdem alle Rechnungen annotiert wurden, können die Annotationen im `json`-Format über den 'Export'-Button exportiert werden.

Listing 7: Beispiel für eine exportierte json-Datei

```
{
    "company-name": "Unbekanntes Unternehmen KG",
    "company-address": "Musterstrasse 1",
    "invoice-id": "INV0001",
    "invoice-date": "Nov 28th, 2022",
    "item-header": "Item",
    "qty-header": "HRS/QTY",
    "rate-header": "Rate",
    "total-header": "Subtotal",
    "total-value": "EUR 553.16"
}
```

Invoice Reader

Der Invoice Reader ermöglicht das Extrahieren von Daten aus einer pdf-Datei, daraufhin werden diese Daten im Interface dargestellt.

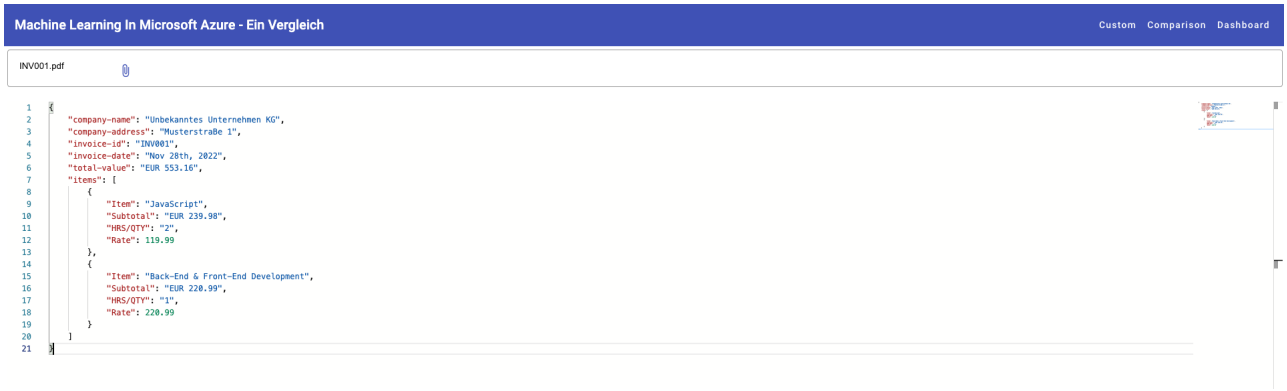

Abbildung 14: Invoice Reader mit Beispieldaten

Nachdem der User eine pdf-Datei auswählt, wird ein HTTP-Request an das Backend abgeschickt und als Antwort wird ein json-Objekt zurückgeliefert, welches die extrahierten Daten enthaltet. Diese Daten werden daraufhin formatiert und in einem monaco-Editor dargestellt.

3.3 Backend

Um die Funktionen im Frontend mit den benötigten Daten zu beliefern, wurden für das Backend drei Azure Functions erstellt, welche mit HTTP-Requests aufrufbar sind. Dazu wurde ein NER-Modell mithilfe von einem Notebook konfiguriert und antrainiert.

3.3.1 NER-Modell

spaCy ist die marktführende Technologie im Bereich NLP und bietet die Möglichkeit an ein eigenes NER-Modell anzutrainieren. Um dieses Modell anzutrainieren, müssen die Daten einem bestimmten Format entsprechen. Dieses Format entspricht einem Array, welches aus Tuples besteht. Das erste Element ist der Text und das zweite ist die dazugehörigen Entitäten mit Startindex und Endindex.

Listing 8: Beispiel für ein Array mit Trainingsdaten

```
1     [
2         ("Tokyo Tower is 333m tall.", [(0, 11, "BUILDING")]),
3     ]
```

Um die vom Frontend exportierten Felder verarbeiten zu können, müssen sie zuerst in dieses Format gebracht werden. Dafür wird zunächst der Text aus der dazugehörigen Rechnung ausgelesen. Als Nächstes wird für jede Entität der Startindex und Endindex ausgelesen, wobei hier zu beachten ist, dass von OCR erkannter Text nicht immer zu 100% stimmt. Daher wurden drei Rückfallmethoden eingebaut.

1. Der Inhalt der Entität wird genau wie angegeben im Text gesucht.
2. Der Inhalt der Entität wird genau wie angegeben im Text gesucht, jedoch darf ein Zeichen unterschiedlich sein.
3. Der Inhalt der Entität wird genau wie angegeben im Text gesucht, jedoch werden manche Zeichen mit ähnlich aussehenden Zeichen ausgetauscht.

Listing 9: Rückfallmethode 3

```
1     def search_with_similar_chars(str, text, entity):
2         temp_value = ''
3         for i in range(0,len(str)):
4             similar_chars = similar_looking_chars(str[i])
5             if similar_chars:
6                 temp_value += '('+'|'.join(similar_chars)+')'
7             else:
8                 temp_value += str[i]
9         catch_total = re.search(temp_value, text)
10        if catch_total != None:
11            ent_tup = (catch_total.span()[0], catch_total.span()[1], entity)
12            return ent_tup
```

Wie in Listing 9 zu sehen ist, wird mit Regex gearbeitet, was die Suche um einiges vereinfacht. Dabei wird auf den Operator "|" zurückgegriffen, durch den man mehrere unterschiedliche Zeichen für einen einsetzen kann.

Neben den vorgegebenen Entitäten gehören noch die Tabellen-Header zu den verpflichtenden Entitäten, da diese später für die Tabellenextraktion benötigt wird.

Listing 10: Verpflichtende Entitäten

```
1       field_names = {'company-name', 'company-address', 'invoice-id',
            'invoice-date', 'total-value', 'item-header', 'qty-header', 'rate-header',
            'total-header'}
```

Nachdem die Trainingsdaten im richtigen Format sind, kann das Modell antrainiert werden. Dafür wird ein leeres spaCy-Modell geladen und alle Entitäten werden hinzugefügt. Zur Kontrolle wird noch zusätzlich überprüft, ob der Startindex und Endindex wirklich eine vollendete Wortgruppe angeben. Falls dies nicht der Fall ist, wird die Entität übersprungen. Am Ende wird das Modell in einen Ordner exportiert, welches vor der Nutzung noch fertig trainiert werden muss.

Listing 11: Trainieren vom Modell

```
1       nlp = spacy.blank("en")
2       doc_bin = DocBin()
3
4       for training_example in training_data:
5           text = training_example['text']
6           labels = training_example['entities']
7           doc = nlp.make_doc(text)
8           ents = []
9           for start, end, label in labels:
10              span = doc.char_span(start, end, label=label,
                    alignment_mode="contract")
11              if span is None:
12                  print("Skipping entity", label)
13              else:
14                  ents.append(span)
15          filtered_ents = filter_spans(ents)
16          doc.ents = filtered_ents
17          doc_bin.add(doc)
18
19          doc_bin.to_disk("model/training_data.spacy")
```

Utils

Um Codewiederholung zu vermeiden und ähnliche Funktionen zu gruppieren, wurden drei UtilsKlassen erstellt, die später in den Azure Functions aufgerufen werden.

BoundingBoxes Diese UtilsKlasse erkennt jegliche Boxen, welche eine Wortgruppe umranden und gibt diese an den Aufrufer zurück.

Dafür wird zuerst mithilfe von `pytesseract` für jedes Wort oder jede Zahl eine Box gespeichert. Jedoch werden dadurch zusammengehörige Wortgruppen voneinander

getrennt. Um diese zu verbinden, werden zuerst alle Wörter nach Zeilen gruppiert und durchiteriert. Dabei wird der horizontale Abstand zwischen den Boxen berechnet und liegt dieser Wert unter einem gewissen Wert, werden diese zwei Boxen miteinander verbunden.

Listing 12: Syntax um zusammengehörige Wortgruppen zu verbinden

```
    def __merge(self, boxes, dist_limit = 10, cell_threshold=50):
        rows = self.__get_rows(boxes, cell_threshold)
        for row in rows.values():
            i=0
            while i<len(row)-1:
                if self.__calc_horizontal_distance(row[i], row[i+1]) <= dist_limit:
                    row[i] = self.__merge_boxes(row[i], row[i+1])
                    row.pop(i+1)
                else:
                    i += 1
        return [item for sublist in list(rows.values()) for item in sublist]
```

Text Diese UtilsKlasse extrahiert den Text eines Bildes sowohl formatiert als auch unformatiert. Dabei nutzen beide Funktionen pytesseract, um den Text zu lesen und jegliche Informationen zur Position zu gewinnen. Der einzige Unterschied ist, dass der Text auf eine andere Weise aufbereitet wird.

```
                                                    Invoice
FROM                                                TO
HTL Leonding                                        Unbekanntes  Unternehmen  KG
LimesstraBe 12/14                                   Max Mustermann
4060 Leonding                                       MusterstraBe 1
Austria                                             Austria
invoice@htl-leonding.ac.at                          max@mustermann.at
Invoice No.: INV014

                                                    Due:  Jan 4th, 2023
Invoice Date: Nov 28th, 2022
 Item                                HRS/QTY        Rate       Tax              Subtotal
 Hardware Upgrade                      90          500.00     Ust(20%)        EUR  45,000.00
                                                              Invoice Summary
                                                   Subtotal                     EUR 45,000.00
                                                   Ust(20%)                      EUR 9,000.00
                                                   Total                        EUR 54,000.00
```

Abbildung 15: Beispiel für einen formatierten Text

```
Invoice
FROM TO
HTL Leonding Unbekanntes Unternehmen KG
LimesstraBe 12/14 Max Mustermann
4060 Leonding MusterstraBe 1
Austria Austria
invoice@htl-leonding.ac.at max@mustermann.at
Invoice No.: INV014
Due: Jan 4th, 2023
Invoice Date: Nov 28th, 2022
Item HRS/QTY Rate Tax Subtotal
Hardware Upgrade 90 500.00 Ust(20%) EUR 45,000.00
Invoice Summary
Subtotal EUR 45,000.00
Ust(20%) EUR 9,000.00
Total EUR 54,000.00
```

Abbildung 16: Beispiel für einen unformatierten Text

Wie in den überliegenden Abbildungen zu sehen ist, befindet sich der Text in der gleichen Reihenfolge und der einzige Unterschied liegt im Abstand zwischen den Wörtern.

InvoiceData Diese UtilsKlasse extrahiert aus einem übergebenen Text die vorgegebenen Entitäten und Rechnungszeilen. Dafür ist sie in zwei Schritte unterteilt.

Entität Extrahierung Mithilfe vom davor antrainiertem NER-Modell wird der Text verarbeitet und alle Entitäten werden in einem Dictionary gespeichert.

Listing 13: Extrahierung von Entitäten

```
1       nlp_ner = spacy.load(modelPath)
2       doc = nlp_ner(text)
3       for ent in doc.ents:
4           op_dict[ent.label_] = ent.text
```

Tabellen Extrahierung Mit dem davor extrahierten Tabellen-Header wird die Tabellen-Header Zeile bestimmt und der formatierte Text wird bis zur dieser Zeile gekürzt. Mithilfe von pandas kann der übrig gebliebene Text in ein Dataframe umgewandelt werden.

Listing 14: Tabellen Extrahierung

```
    header_row = max(set(header_rows), key=header_rows.count)
    table = formatted_text_lines[header_row:]
    text = '\n'.join(table)

    df = pd.read_csv(StringIO(text), sep='\s{3,}', engine='python',
        index_col=False).dropna()

    items = []
    for i in range(0, df.shape[0]):
        item = {}
        for header_field in header_fields:
            item[op_dict.get(header_field)] = df.loc[i][op_dict.get(header_field)]
        items.append(item)
    op_dict['items'] = items
```

Azure Functions

Um die in Utils definierten Methoden im Frontend aufrufen zu können, wurden sie als Azure Functions zusammengefasst. Das Resultat sind drei über einen HTTP-Request aufrufbare Funktionen.

Genau wie die Utils-Klassen wurden die Azure Functions mit der Skriptsprache Python entwickelt. Zusätzlich wurde die Visual Studio Code Erweiterung für Azure Functions genutzt, um diese zu generieren und zu konfigurieren.

Jede Azure Function besteht aus einer Konfigurationsdatei und einer Datei, welche den eigentlichen Code beinhaltet. Dabei gibt die Konfigurationsdatei das Authentifizierungsmittel, die erlaubten HTTP-Methoden und die Auslöseart an. Im unterliegenden Listing wird die Azure Function zum Beispiel mithilfe von einem POST-HTTP-Request ausgelöst.

Listing 15: Beispiel fuer eine Konfigurationsdatei

```
{
    "scriptFile": "__init__.py",
    "bindings": [
        {
            "authLevel": "function",
            "type": "httpTrigger",
            "direction": "in",
            "name": "req",
            "methods": [
                "post"
            ]
        },
        {
            "type": "http",
            "direction": "out",
            "name": "\$return"
        }
    ]
}
```

Die drei Azure Functions wurde jeweils so aufgeteilt, dass sie eine bestimmte Aufgabe erfüllen.

Bounding Box Wie der Name sagt werden die Bounding Boxes von dieser Azure Function generiert. Dafür wird zuerst das Bild im Byte-Format aus dem HTTP-Request-Body in ein von Python verarbeitbares Image umgewandelt. Als Nächstes wird die Utils-Klasse initialisieren und zum Generieren der BoundingBoxes genutzt.

Das Result wird in einem HTTP-Response-Body zurückgegeben, dabei hat der Request den Response Code 200, welcher angibt, dass der Request erfolgreich durchgeführt wurde.

In zwei Situationen werden Response Codes zurückgegeben, die indizieren, dass der Request nicht fehlerfrei umgesetzt wurde. Diese Wären:

1. Falls der HTTP-Request-Body leer ist, wird ein Response Code von 400 zurückgegeben, der angibt, dass der HTTP-Request nicht valide aufgebaut ist.
2. Falls beim Generieren ein Fehler auftritt, wir ein Response Code von 500 zurückgegeben, der angibt, dass serverseitig ein Fehler aufgekommen ist.

Size Da das Frontend die pdf-Datei in einer abweichenden Grüße darstellt, wird die Originalgröße benötigt, um die Bounding Boxen auf der richtige Position darzustellen. Diese Bildgröße wird in einem Tuple zurückgegeben.

Die Response Codes sind ident zu den von der Bounding Box Azure Function.

Invoice Data Damit das Modell von außen erreichbar ist, dient diese Azure Function als Schnittstelle. Beim Aufruf der Methode besteht die Möglichkeit, den Pfad zum gespeicherten Modell anzugeben. Da spaCy sowohl das präziseste als auch das in der letzten Iteration erstellte Modell speichert, hat man die Möglichkeit zwischen eines der beiden zu entscheiden. Normalerweise sollte das präziseste Modell genutzt werden, da man mit diesem Modell auch das beste Ergebnis erhaltet.

Listing 16: Invoice Data Azure Function

```
1       def main(req: func.HttpRequest) -> func.HttpResponse:
2           try:
3               logging.info('Invoice data registered a request.')
4
5               body = req.get_body()
6               pages = convert_from_bytes(body)
7               image = np.array(pages[0])
8
9               logging.info('Bytes have been successfully converted into a Image.')
10
11              if body:
12                  invoice_data = InvoiceData()
13
14                  logging.info('InvoiceData class has been initialized.')
15
16                  data = invoice_data.get_invoice_data(image, 'model/model-best')
17
18                  logging.info('Invoice data has been successfully read.')
19
20                  return func.HttpResponse(json.dumps(data),
                        mimetype="application/json")
21              else:
22                  return func.HttpResponse('Request does not contain a body.',
                        status_code=400)
23          except Exception as e:
24              logging.exception(e)
25              return func.HttpResponse("An error occurred while processing the
                    request", status_code=500)
```

Wie man im Listing 16 sieht, werden die wichtigsten Informationen mithilfe von einem Logger ausgegeben. Dies hilft beim Auftreten von einem Fehler, dass dieser Fehler schnellstmöglich lokalisiert und behoben wird.

4 Zusammenfassung

4.1 Gelerntes

Dadurch, dass diese Diplomarbeit mein erster Kontakt mit OCR, NER und generell Informationsextraktion war, wurde die meiste Zeit in die Recherche gesteckt. Während der Recherche bin ich auf unterschiedliche Ansätze gestoßen, davon wurden die meisten auch ausgetestet und gleich wieder verworfen, da sie für unseren Anwendungsfall nicht geeignet sind.

Ein Beispiel dafür war die Tabellenextraktion, bei der drei unterschiedliche Ansätze getestet wurden. Als Erstes wurden Libraries getestet, welche von sich selber die Möglichkeit anbieten, eine ganze Methode zu extrahieren. Jedoch waren diese nicht passend, da unsere Beispieldaten nicht herkömmlich formatiert sind. Das gleiche Problem war der Grund, dass die komplett selbst entwickelte Tabellenextraktion verfallen wurde.

Außerdem wurde beim Antrainieren des Modells getestet, was die minimale Anzahl der Trainingsdaten ist. Daher wurde das erste Modell mit nur fünf Rechnungen trainiert, wo auch schnell klar wurde, dass dies viel zu wenig sind, da zum Beispiel die Adressen nur zum Teil extrahiert wurden und manche Feld komplett falsch kategorisiert wurden. Daraufhin wurde die Anzahl der Trainingsdaten stufenhaft erhöht und am Ende wurde kam die optimale Anzahl auf rund 15 Rechnungen.

Abgesehen vom technischen Teil wurde mir auch organisatorisch viel für die Zukunft mitgegeben. Da ich im Vorfeld nicht damit gerechnet habe, dass man mehrere unterschiedliche Ansätze austesten muss, bis man den richtigen findet, wurde viel mehr Zeit in die Entwicklung gesteckt, als am Anfang geplant.

4.2 Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Ich bedanke mich bei jedem Mitarbeiter von smartpoint, die mich in jeglicher Art unterstützt haben. Zusätzlich bedanke ich mich bei smartpoint dafür, dass sie für die Diplomarbeit eine Testumgebung zur Verfügung gestellt haben und dass sie auch nach der Praktikumszeit für Fragen ein offenes Ohr hatten.

Außerdem möchte ich mich bei Nico Bojer für das mehrmalige Korrekturlesen meiner Diplomarbeit danken.

Weiters gilt mein Dank meinem Diplomarbeitsbetreuer, Karpowicz Michał, welcher mich auch in stressigen Situationen unterstützt hat.

Glossar

AI Artificial Intelligence

CLI Command Line Interface

CSS Cascading Style Sheets

DL Deep Learning

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ICDAR International Conference of Document Analysis and Recognition

KI Küntliche Intelligenz

ML Machine Learning

MSER Maximally Stable Extremal Regions

NER Named Entity Recognition

NLP Natura Language Processing

OCR Optical Character Recognition

SCSS Sassy Cascading Style Sheets

SWT Stroke Width Transformer

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

Literaturverzeichnis

[1] W. P. u. A. S. u. J. S. u. M. T. A. T. Peter Stone und Rodney Brooks und Erik Brynjolfsson und Ryan Calo und Oren Etzioni und Greg Hager und Julia Hirschberg und Shivaram Kalyanakrishnan und Ece Kamar und Sarit Kraus und Kevin Leyton-Brown und David Parkes, „Artificial Intelligence and Life in 2030." 2016. Online verfügbar: https://ai100.stanford.edu/sites/g/files/sbiybj18871/files/media/file/ai100report10032016fnl_singles.pdf

[2] N. J. Nilsson, *The Quest for Artificial Intelligence*. Cambridge, UK: Cambridge University Press, 2009.

[3] B. B. und Daniel Klemm und Dr. Carlo Velten, „Machine Learning im Unternehmenseinsatz - Künstliche Intelligenz als Grundlage digitaler Transformationsprozesse," 2017. Online verfügbar: https://cloudflight.io/app/uploads/2022/02/studie_machine-learning-im-unternehmenseinsatz.pdf

[4] D. E. Ullmann, „Lernen aus neurobiologischer Perspektive," 2015. Online verfügbar: https://www.vhs-st.de/fileadmin/user_upload/Eigene_Daten/Projekte/PUYB/OEsterreich/20161006_WS_04_Neurobiologie.pdf

[5] „Entwicklung von Gehirn und Nervensystem," letzter Zugriff am 12.09.2022. Online verfügbar: https://www.neurologen-und-psychiater-im-netz.org/gehirn-nervensystem/entwicklung

[6] J. Delua, „Supervised vs. Unsupervised Learning: What's the Difference? | IBM," 2021, letzter Zugriff am 03.27.2022. Online verfügbar: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

[7] A. Biswal, „The Complete Guide on Overfitting and Underfitting in Machine Learning," 2021, letzter Zugriff am 13.09.2022. Online verfügbar: https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting

[8] C. N. N. und Oliver Zeigermann, *Machine Learning kurz und gut*, 1. Aufl. dpunkt.verlag, 2018.

[9] L. Wuttke, „Training-, Validierung- und Testdatensatz," 2021, letzter Zugriff am 01.04.2022. Online verfügbar: https://datasolut.com/wiki/trainingsdaten-und-testdaten-machine-learning/

[10] J. JORDAN, „Evaluating a machine learning model." 2017, letzter Zugriff am 13.09.2022. Online verfügbar: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/

[11] L. Eikvil, „OCR - Optical Character Recognition," 1993, letzter Zugriff am 03.27.2022. Online verfügbar: http://home.nr.no/~eikvil/OCR.pdf

[12] G. Shperber, „A gentle introduction to OCR." 2018, letzter Zugriff am 03.27.2022. Online verfügbar: https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa

[13] M. Opitz, „Text Detection and Recognition in Natural Scene Images," 2018, letzter Zugriff am 03.27.2022. Online verfügbar: https://cvl.tuwien.ac.at/wp-content/uploads/2014/12/tr-11.pdf

[14] B. E. und Eyal Ofek und Yonatan Wexler, *Detecting Text in Natural Scenes with Stroke Width Transform.* Microsoft Corporation, 2010, letzter Zugriff am 01.04.2022. Online verfügbar: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/201020CVPR20TextDetection.pdf

[15] „ICDAR," letzter Zugriff am 03.30.2022. Online verfügbar: https://icdar2021.org

[16] „Natural Language Processing," 2001, letzter Zugriff am 01.12.2022. Online verfügbar: https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub

[17] „Natural Language Processing (NLP)," 2020, letzter Zugriff am 01.12.2022. Online verfügbar: https://www.ibm.com/cloud/learn/natural-language-processing

[18] C. Marshall, „What is named entity recognition (NER) and how can I use it?" 2019, letzter Zugriff am 04.12.2022. Online verfügbar: https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d

[19] V. Meel, „Artificial Neural Network: Everything you need to know," letzter Zugriff am 12.09.2022. Online verfügbar: https://viso.ai/deep-learning/artificial-neural-network/

[20] L. Wuttke, „Machine Learning vs. Deep Learning: Wo ist der Unterschied?" letzter Zugriff am 12.09.2022. Online verfügbar: https://datasolut.com/machine-learning-vs-deep-learning

[21] „HTML & CSS," letzter Zugriff am 14.11.2022. Online verfügbar: https://www.w3.org/standards/webdesign/htmlcss

[22] „HTML/Tutorials/Entstehung und Entwicklung," letzter Zugriff am 14.11.2022. Online verfügbar: https://wiki.selfhtml.org/wiki/HTML/Tutorials/Entstehung_und_Entwicklung

[23] „HTML Attributes," letzter Zugriff am 17.11.2022. Online verfügbar: https://www.w3schools.com/html/html_attributes.asp

[24] „Übersicht: Attribute der HTML-Tags," 2022, letzter Zugriff am 17.11.2022. Online verfügbar: https://www.mediaevent.de/html/HTML-Attribute.html

[25] „Die 10 wichtigsten HTML-Tags," 2018, letzter Zugriff am 14.11.2022. Online verfügbar: https://code-crowd.de/blog/die-10-wichtigsten-html-tags/

[26] „HTML <!DOCTYPE> Declaration," letzter Zugriff am 14.11.2022. Online verfügbar: https://www.w3schools.com/tags/tag_doctype.asp

[27] „CSS/Tutorials/Einstieg/Stylesheets einbinden," letzter Zugriff am 22.11.2022. Online verfügbar: https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Stylesheets_einbinden#Einbindung

[28] „Sass Basics," letzter Zugriff am 22.11.2022. Online verfügbar: https://sass-lang.com/guide

[29] „2021 Developer Survey," 2021, letzter Zugriff am 13.09.2022. Online verfügbar: https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies

[30] „Displaying values with interpolation," 2022, letzter Zugriff am 13.11.2022. Online verfügbar: https://angular.io/guide/interpolation

[31] „The TypeScript Handbook," 2022, letzter Zugriff am 04.12.2022. Online verfügbar: https://www.typescriptlang.org/docs/handbook/intro.html

[32] G. van Rossum, „Foreword for "Programming Python"(1st ed.)," 1996, letzter Zugriff am 01.04.2022. Online verfügbar: https://www.python.org/doc/essays/foreword/

[33] „Intro to data structures," letzter Zugriff am 04.12.2022. Online verfügbar: https://pandas.pydata.org/docs/user_guide/dsintro.html#:~:text=DataFrame%20is%20a%202%2Ddimensional,most%20commonly%20used%20pandas%20object.

[34] A. A. P. und Ajay Uppili Arasanipalai, *Applied Natural Language Processing in the Enterprise*, 1. Aufl. O'Reilly, 2021.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis