

Abschlussprüfung Winter 2020/2021

FACHINFORMATIKER ANWENDUNGSENTWICKLUNG

DOKUMENTATION ZUR BETRIEBLICHEN PROJEKTARBEIT

Klassifizierung von E-Mails

Mithilfe von Künstlicher Intelligenz (im weiteren Verlauf genannt: „KI“)

Abgabe: Dortmund 25.10.2021

Prüfungsbewerber:

PHILIPP BRAUN

WINKELGASSE 8, 58642 ISELOHN

Ausbildungsbetrieb:

QUALIFIZIERUNGS AKADEMIE RHEINRUHR GMBH & CO. KG

STOCKHOLMER ALLEE 30 C, 44269 DORTMUND

Praktikumsbetrieb:



SYLVENSTEIN MEDIA GMBH

KARL-ZAHN-STRASSE 11, D-44141 DORTMUND, GERMANY

TEL: +49 (0) 231-95 25 354

FAX: +49 (0) 231-95 25 45

Inhaltsverzeichnis

1	<u>EINLEITUNG</u>	6
1.1	EINLEITUNG	6
1.2	PROJEKTUMFELD	6
1.2.1	BETRIEBLICHE LERNPHASE: SYLVENSTEIN GMBH	6
1.2.2	KUNDE	6
1.3	PROJEKTZIEL	6
1.4	PROJEKTSCHNITTSTELLEN	7
1.4.1	TECHNISCH	7
1.4.2	PERSONELL	7
2	<u>ANALYSE</u>	7
2.1	VORGESPRÄCH	7
2.1.1	PROJEKTBEGRÜNDUNG	7
2.2	IST-ANALYSE	8
2.3	SOLL KONZEPT	8
2.3.1	GRUNDKONZEPT	8
2.3.2	PROJEKTABGRENZUNG	9
2.3.3	SPAM ERKENNEN	9
2.3.4	TAG DES THEMENGEBIETES SETZEN	10
2.3.5	TEST- UND PRODUKTIV-SYSTEM	10
2.3.6	E-MAIL ZUM ZUSTÄNDIGEN MITARBEITER VERSCHIEBEN	10
2.4	WIRTSCHAFTLICHKEITSANALYSE	11
3	<u>PROJEKTPLANUNG</u>	12
3.1	ERSTELLUNG DES LASTEN- UND PFLICHTENHEFTES	12
3.1.1	LASTENHEFT	12
3.1.2	PFLICHTENHEFT	12
3.2	ERFASSUNG DER ARBEITSPAKTE MIT ZUORDNUNG ZU DEN PROJEKTPHASEN	13
3.3	PROJEKTPHASEN MIT ZEITPLANUNG	13
3.4	ZEITLICHE PLANUNG	14
3.5	RESSOURCENPLANUNG	15

3.6	ENTWICKLUNGSPROZESS	15
3.7	KOSTEN-PLANUNG	16
3.7.1	MAKE OR BUY ENTSCHEIDUNG	16
3.7.2	PROJEKTKOSTEN	17
3.7.3	AMORTISIERUNG	17
3.7.4	KUNDE	17
3.7.5	ABGRENZUNG: "REINE WIRTSCHAFTLICHKEIT"	19
3.7.6	ENTWICKLUNG	19
4	DESIGN/ENTWURFS-PHASE	19
4.1	ENTWURF DATENBANK & SCHNITTSTELLEN	19
4.1.1	PLATTFORM: COLAB	19
4.1.2	DATENBANK	19
4.1.3	PROGRAMMIERSPRACHE UND BIBLIOTHEKEN	21
4.1.4	SCHNITTSTELLEN	21
4.1.5	TEXTVORVERARBEITUNG (TEXTPREPROCESSING)	23
4.2	DEFINITION DER KLASSIFIZIERUNG FÜR SPAM & THEMENBEREICHE	27
4.2.1	EINLEITUNG	27
4.2.2	UNLABELED DATA (UNGEKENNZEICHNETE DATEN)	27
4.2.3	LABELD DATA (GEKENNZEICHNETE DATEN)	28
4.3	FESTLEGUNG VON KEYWORDS UND DEN DAZUGEHÖRIGEN THEMENBEREICHEN	28
4.4	DESIGN DES NEURONALES NETZ	28
4.4.1	SUPERVISED MACHINE LEARNING (ÜBERWACHTES LERNEN)	28
4.4.2	NEURONALE NETZE	28
4.4.3	DESIGN/ENTWURF NEURONALES NETZWERK	29
5	PROJEKT DURCHFÜHRUNG	29
5.1	SAMMELN UND ORDNEN VON DEMODATEN	29
5.1.1	EXPORT VON EMAILS	29
5.1.2	OUTLOOK	30
5.1.3	CSV	30
5.1.4	TEST-SYSTEM	30
5.1.5	COLAB	30
5.2	TRAININGSDATEN AUFBEREITEN UND IMPORTIEREN	30

5.2.1	MySQL	30
5.2.2	DATENKENNZEICHNUNG(DATA LABELING)	31
5.3	ANPASSEN VON THEMENGEBIETEN MIT SCHLÜSSELWORTEN ZUR WEITEREN KLASSIFIZIERUNG	31
5.3.1	KLASSIFIZIERUNG DER THEMENGEBIETE	31
5.4	TEXTVORVERARBEITUNG & SCHLÜSSELWORT EXTRAKTION	32
5.4.1	TEXTANREICHERUNG	32
5.4.2	GERÄUSCHENTFERNUNG (NOISE REMOVAL)	33
5.4.3	NORMALISIERUNG	33
5.4.4	ABGRENZUNG: PROJEKTANTRAG	36
5.5	KLASSIFIZIERUNG UND AUSWERTUNG DER E-MAILS DURCH KI	36
5.5.1	IMPLEMENTIERUNG DER KI-SYSTEM	36
5.6	ITERATIVES TRAINING DER KI-MODELLE UND ANPASSUNG DES NEURONALEN NETZES	39
5.6.1	TRAINIEREN	39
6	TEST-PHASE	42
6.1	VERGLEICH IST UND SOLL DER KLASSIFIZIERUNG	43
6.1.1	ÜBERSCHNEIDUNG: "ITERATIVES TRAINIEREN"	43
6.1.2	ABSCHLUSS BETRACHTUNG DES TRAININGS	43
6.1.3	AUSWERTUNG	43
6.1.4	SOLL/IST VERGLEICH	43
6.2	TEST-SZENARIEN MIT TRAININGSDATEN (BLACK- & WHITE-BOX-TEST)	44
6.2.1	TEST-SYSTEM (WHITE-BOX-TEST)	44
6.2.2	PRODUKTIV-SYSTEM(BLACK-BOX-TEST)	44
7	PROJEKTABSCHLUSS	44
7.1	ÜBERGABE DES PROJEKTES AN DEN KUNDEN	44
7.1.1	ÜBERGABE	44
7.2	FAZIT	45
7.2.1	AUSBLICK	45
7.2.2	WISSENSERWERB (KNOW HOW)	45
7.2.3	SPEICHERN UND LADEN VON KI-MODELLEN	46
7.2.4	DEPLOY MODELL IN COLAB VIA FASTAPI	46
7.2.5	WEBHOSTING MIT FASTAPI	46
7.3	ERSTELLEN DER DOKUMENTATION	46

8 ANHANG	I
8.1 LITERATURVERZEICHNIS	I
8.2 TABELLENVERZEICHNIS	I
8.3 ABBILDUNGSVERZEICHNIS	I
8.4 GLOSSAR	II
8.4.1 SCRUM	II
8.4.2 SIGMOID-FUNKTION	III
8.4.3 CSV	III
8.4.4 TEXTBLOB	III
8.4.5 SENTIMENT-ANALYSE	III
8.4.6 COLABORATORY	III
8.4.7 OBJEKTSERIALISIERUNG	IV
8.4.8 NAIVE BAYES	IV
8.4.9 LINEARE REGRESSION	IV
8.4.10 LOGISTISCHE REGRESSION	V
8.4.11 SUPPORT-VEKTOR-MASCHINE (SVM)	V
8.4.12 K-NÄCHSTER NACHBAR(K-NEAREST NEIGHBOR)	VI
8.4.13 ZUFÄLLIGER WALD (RANDOM FOREST)	VI
8.5 ABKÜRZUNGSVERZEICHNIS	VI
8.6 LITERATURVERZEICHNIS	VI
8.6.1 TABELLEN & ABBILDUNGEN	VII
DOKUMENTEN ANHANG	IX

1 Einleitung

1.1 Einleitung

Die folgende Dokumentation wurde als Teil der Abschlussprüfung der IHK Dortmund für Ausbildung zum Fachinformatiker für Anwendungsentwicklung erstellt. Christian Giebel ist Geschäftsführer und Ausbilder der Sylvenstein GmbH. In diesem Projekt soll es darum gehen eintreffende unerwünschte E-Mails (Spam) durch ein KI-System als Spam zu klassifizieren. Außerdem soll eine automatische Zuordnung der E-Mail anhand ihres Inhaltes zum zuständigen Mitarbeiter stattfinden.

1.2 Projektumfeld

1.2.1 Betriebliche Lernphase: Sylvenstein GmbH

Das Unternehmen Sylvenstein GmbH hat seinen Sitz in Dortmund. Sylvenstein ist ein junges IT-Unternehmen, welches sich auf Software-Lösungen für Bildungsanbieter spezialisiert hat. Das Unternehmen besteht aus einem Geschäftsführer, drei festangestellten Entwicklern, zwei externen Entwicklern und drei Auszubildenden im ersten und zweiten Lehrjahr.

1.2.2 Kunde

Der Kunde dieses Projektes ist das Unternehmen Sylvenstein. Durch dieses Projekt sollen verwaltungstechnische Arbeitsabläufe im Bereich der firmeninternen Organisation automatisiert und verbessert werden. Außerdem übernimmt dieses Projekt Aufgaben eines Sekretärs, der für den elektronischen Posteingang zuständig ist. Ebenfalls soll dieses Projekt auch einen Wissenserwerb(Knowhow) im Bereich der künstlichen Intelligenz(KI) darstellen und eine Grundlage für zukünftige Projekte in diesem Bereich sein.

1.3 Projektziel

Eintreffende E-Mails sollen von der KI auf Spam oder nicht Spam geprüft werden. Spam-E-Mails sollen aussortiert werden. Anschließend soll die KI auch den Inhalt der E-Mail auswerten und daraufhin, das Thema der Mail vorhersagen. Das von der KI vorausgesagte Thema soll dann die Grundlage sein, damit eine weitere Zuordnung der E-Mail zu Mitarbeiter stattfinden kann.

1.4 Projektschnittstellen

1.4.1 Technisch

Für dieses Projekt ist eine Schnittstelle zu Semi¹ und dem damit verbundenen Mail- und Datenbank-Server nötig. An Semi wird zur Entwicklung eine MySQL-Datenbank angeschlossen, in der die Daten der Emails, und die Zuständigkeiten der Mitarbeiter in Tabellen abgelegt werden. Die KI-Systeme werden in Colaboratory² (Colab) ausgeführt und trainiert. Zentrale Schnittstelle zwischen der Datenbank und den KI-Systemen ist die Python-Bibliothek SQLAlchemy³. Diese Bibliothek ermöglicht eine verschlüsselte Verbindung zwischen MySQL, Semi und Colab, um die Daten aus MySQL in Colab zu importieren und nach der Auswertung wieder zurück in die Datenbank zu exportieren. Danach soll Semi ebenfalls auf die Datenbank zugreifen, dort die Vorhersagen für die E-Mail auslesen und auf Grundlage dessen die E-Mails im Mail-Server verschieben.

1.4.2 Personell

Die Mitarbeiter von Sylvenstein bilden mit ihren Zuständigkeiten die Grundlage für die Mitarbeitertabelle. Intern wurde das Projekt von Christian Giebel als Ausbilder und Projektleiter betreut.

2 Analyse

2.1 Vorgespräch

Zu Beginn wurde ein Vorgespräch mit dem Ausbilder Christian Giebel geführt. Der besondere Personal- bzw. Zeitaufwand für das Verwalten des E-Mail-Eingangs ist der Anlass, der zur Projektidee führte.

2.1.1 Projektbegründung

Durch das Projekt wird erwartet, dass die Kommunikation mit den Kunden verbessert wird. Die Kommunikation mit den Kunden soll dadurch verbessert werden, dass der allgemeine Posteingang von der KI verwaltet werden kann. Unerwünschte E-Mails (Spam) sollen erkannt und aussortiert werden. Dadurch sollen die Anfragen der Kunden schneller dem zuständigen Mitarbeiter zugewiesen und bearbeitet werden können. Außerdem reduziert sich der Arbeitsaufwand, da das allgemeine Postfach von keinem Mitarbeiter mehr überwacht werden muss. Dieser

¹ Semi ist eine eigene Software von Sylvenstein zur Seminarverwaltung und verfügt über Module zur Anbindung und Verwaltung eines Mail-Servers, diese Module werden in diesem Projekt verwendet.

² Siehe: 4.1.1 Plattform Colab & 8.4.6 Colaboratory

³ Siehe: 4.1.4.1.1 SQL-Alchemy

Arbeitsaufwand hätte sonst langfristig Entwickler-Ressourcen für verwaltungstechnische Aufgaben gebunden, was mit dem KI-System vermieden werden kann. Zusätzlich ist das Projekt nicht nur von einem rein wirtschaftlichen Standpunkt zu betrachten. Da Sylvenstein in diesem Projekt, erstmalig die neue Technologie im Bereich KI einsetzen will. Genauer gesagt soll in diesem Projekt die einfachste Form des maschinellen Lernens verwendet werden (Überwachtes Lernen, Supervised Learning), um einen Wissenserwerb im Bereich KI anzustoßen. Eine weitere interne Begründung für diese Projekt ist die Wiederverwendbarkeit des erworbenen Wissens und der modulare Ansatz der Programmierung. Im Idealfall ergeben sich weitere Anwendungsbereiche im Bereich KI aus diesem Projekt.

2.2 Ist-Analyse

Es werden täglich Mitarbeiter-Ressourcen dadurch gebunden, dass der allgemeine Posteingang nicht verwaltet wird und unerwünschte E-Mails eintreffen. Jeder Mitarbeiter muss täglich mehrmals das Postfach überprüfen, ob neue Nachrichten eingegangen sind, die ihn betreffen. Sylvenstein hat geplant einen Sekretär in Teilzeit einzustellen, der das Postfach verwalten würde. Diese Stelle würde entfallen bei erfolgreicher Umsetzung des Projektes.

2.3 Soll Konzept

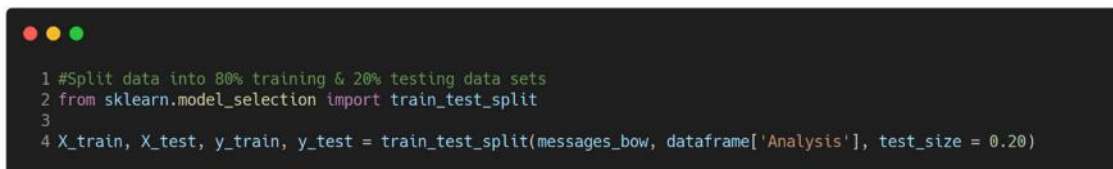
2.3.1 Grundkonzept

Es sollen eintreffende E-Mails aus dem allgemeinen Postfach auf Spam überprüft werden. Außerdem soll der Inhalt der E-Mail analysiert werden und darauf in das persönliche Postfach des entsprechenden Mitarbeiters verschoben werden. Anhand der Zuständigkeiten der Mitarbeiter von Sylvenstein soll in einer Datenbank eine Tabelle erstellt werden, in der aufgelistet wird, welcher Mitarbeiter für welche Themengebiete zuständig ist und welche Tags dann diesen Themen zugeordnet werden. Diese Tabelle dient als Grundlage für die spätere Zuordnung der E-Mails. Es folgt eine beispielhafte Darstellung. Aus Datenschutzgründen werden hier nur Beispieldaten genannt. Anhand der Themen „Gehe“ und „Xamarin“ kann eine Zuordnung zum entsprechenden Mitarbeiter stattfinden, der für diese Themen zuständig ist.

Mitarbeiter	Themen	Tags
<i>Name Vorname</i>	<i>Kunde: Gehe; Technologie: Xamarin</i>	<i>Gehe, Xamarin</i>
...

Tabelle 1 – Mitarbeitertabelle

In der Datenbank soll eine weitere Tabelle angelegt werden, in der die E-Mails abgelegt werden sollen, die zum Trainieren der KI benötigt werden. Bei diesen E-Mails handelt es sich um den echten E-Mail-Verkehr der Firma Sylvenstein, diese Daten müssen für das überwachte maschinelle Lernen (supervised machine learning) aufbereitet (Textvorverarbeitung⁴) und gelabelt (gekennzeichnet)⁵ werden. Diese aufbereiteten Daten sollen dann von einem neuronalen Netzwerk verarbeitet werden. Dieses Netz verarbeitet dann die Daten und trifft dann eine Vorhersage für jeden Eintrag im Datensatz. Diese Vorhersage kann dann mit dem Label (Kennzeichnung) der Daten verglichen werden. Trifft die KI eine falsche Vorhersage, kann das neuronale Netz für den nächsten Durchlauf (Iteration) angepasst werden. Grundlage für diese Art des maschinellen Lernens ist das „Trial-and-Error“-Prinzip, was nach jedem Fehler zu einer Verbesserung und Anpassung des Netzwerkes führt. Die Daten werden in X und Y, und in Trainings- und Testdaten aufgeteilt. „test_size“ ist die prozentuale Aufteilung der Daten zwischen Trainings- und Testdaten.



```
1 #Split data into 80% training & 20% testing data sets
2 from sklearn.model_selection import train_test_split
3
4 X_train, X_test, y_train, y_test = train_test_split(messages_bow, dataframe['Analysis'], test_size = 0.20)
```

Abbildung 1 – Trainingsdaten aufteilen

2.3.2 Projektabgrenzung

Das Projekt dieser Dokumentation ist geplant als Teilstück der firmeninternen Verwaltungs-Software und der Seminar-Verwaltungs-Software (Semi). Dieses Projekt soll dann als Modul eingesetzt werden können, das zur internen Verwaltung genutzt werden kann. Die KI-Systeme sollen unabhängig arbeiten und modular einsetzbar sein, deshalb wird das Projekt in Python programmiert. Damit ist es möglich neuronale Netze einmal zu entwickeln und an anderer Stelle aufzurufen um Vorhersagen zutreffen (Objektserialisierung⁶). Langfristig könnte diese Technologie dann auch bei zukünftigen Kunden und weiteren Projekten eingesetzt werden.

2.3.3 Spam erkennen

Die Spam-Erkennung der KI soll auch nach diesem Prinzip arbeiten, die KI soll mit den aufbereiteten und gekennzeichneten Daten iterativ trainiert werden, bis die gewünschte Genauigkeit bei der Spam-Erkennung erreicht ist. Die Besonderheit des neuronalen Netzes bei der Spamerkennung ist, dass als Output-Layer ein einzelnes Neuron ausreicht, das einen booleschen

⁴ Erläuterung in Abschnitt: 4.1.6 Textpreprocessing

⁵ Erläuterung in Abschnitt: 4.2.2 Textvorverarbeitung (Textpreprocessing & 4.2.3 Labeld Data (gekennzeichnete Daten)

⁶ Siehe Anhang Glossar: 8.4.7 Objektserialisierung

Wert zurückgibt, je nachdem ob der Inhalt der E-Mail auf Spam oder nicht Spam hindeutet. Zur Visualisierung lassen sich die Daten, die in Spam und nicht Spam Mails aufteilen. Diese Aufteilung führt dazu, dass „Spam-„ und „nicht-Spam-Wörterlisten“ erstellt werden können. Aufgrund dieser Wörterlisten kann das KI-System zukünftig Mails bewerten, indem die Wörter, die im Inhalt erkannt werden, eher den Spam- oder nicht-Spam Wörtern zugeordnet werden können.

2.3.4 Tag des Themengebietes setzen

Die Vorhersagen der Tags, anhand des Inhaltes der E-Mail sollen ebenfalls mit Überwachtem Lernen (Supervised Learning) geleistet werden, aber anders als bei der Spam-Erkennung, sollen hier ein oder zwei oder kein Wert für die Tags zurückgegeben werden. Deshalb wird in diesem KI-System ein anderes neuronales Netzwerk eingesetzt, das über ein anders konfiguriertes Output-Neuron verfügt. Es soll ein Array mit booleschen Werten von der Länge "n x Anzahl Tags" zurückgeben. Es soll nur Tags mit einer Übereinstimmung über einem bestimmten Schwellenwert zurückgegeben. Sobald die KI hier auch die gewünschte Genauigkeit bei der Vorhersage von Tags zum Inhalt der E-Mails erreicht hat, sollen die vorhergesagten Tags Semi dazu dienen, die E-Mails zum zuständigen Mitarbeiter zu verschieben.

2.3.5 Test- und Produktiv-System

Das iterative Training beider KI-Modelle soll in einer Testumgebung stattfinden. In der Testumgebung soll Semi keine E-Mails aus den Trainingsdaten zu Mitarbeitern verschieben. Allerdings soll das Verschieben von E-Mails des Semi-Moduls getestet werden, um einen Einsatz im Produktiv-System gewährleisten zu können. Das Produktiv-System soll eine identische Kopie des Test-Systems sein, um die Modelle einsetzen zu können. Außerdem bietet diese Trennung Sylvenstein die Möglichkeit, im Test-System weitere Daten in der E-Mail und Mitarbeiter Tabelle zu ergänzen und die Modelle weiter zu trainieren und zu verbessern.

2.3.6 E-Mail zum zuständigen Mitarbeiter verschieben

In diesem Abschnitt soll erläutert werden, wie die Zuordnung der E-Mail ablaufen soll. Eintreffende Emails sollen von Semi in die MySQL-Datenbank des KI-Systems exportiert werden, es soll eine Vorverarbeitung des Textes⁷ der E-Mail stattfinden. In dieser Vorverarbeitung sollen bestimmte Stoppwörter⁸ der deutschen und englischen Sprache entfernt werden. Die Zeichensetzung soll normalisiert⁹ und Sonderzeichen entfernt werden.

⁷ Siehe: 4.1.5 Textvorverarbeitung (Textpreprocessing)

⁸ Stoppwörter: <https://de.wikipedia.org/wiki/Stoppwort> Stand: 04.10.2021

⁹ Normalisierung: [https://de.wikipedia.org/wiki/Normalisierung_\(Text\)](https://de.wikipedia.org/wiki/Normalisierung_(Text)) Stand: 04.10.2021

Damit soll die Auswertungsfähigkeit der KI gesteigert werden. Daraufhin soll die KI den vorverarbeiteten Text analysieren und vorhersagen, ob es sich um eine Spam-E-Mail handelt und die passenden Tags setzen. Nach Auswertung der E-Mail durch die KI soll Semi die E-Mail anhand der Vorhersagen verschieben. Spam-E-mails sollen in einen Spam-Ordner verschoben werden. Außerdem sollen Emails, bei denen die Tags mit dem Tag eines Mitarbeiters aus der Mitarbeitertabelle übereinstimmen, direkt zum zuständigen Mitarbeiter verschoben werden. Semi hat vollen Zugriff auf die MySQL-Datenbank und kann die E-Mail anhand der Spalten id, pred_Spam, pred_Tag aus der E-Mail-Tabelle und der Tabelle mit den Zuständigkeiten der Mitarbeiter automatisch die Verwaltung des Mail-Servers übernehmen. Die Programmierung der Funktionsweise von Semi ist nicht Bestandteil dieser Dokumentation und wird aus Datenschutzgründen nicht aufgeführt.

2.4 Wirtschaftlichkeitsanalyse

Vorab, dieser Abschnitt soll dazu dienen, abschätzen zu können, ob sich die Projektidee wirtschaftlich durchführen lässt. Von Sylvenstein steht mir ein Arbeitsplatz mit Windows-PC zur Verfügung gestellt, dessen Gemeinkosten ebenfalls zu berücksichtigen sind. Zur Programmierung der KI wird Colab¹⁰ verwendet. Colab verfügt über eine eigene Laufzeitumgebung und einen Kernel und steht kostenlos im Browser zur Verfügung. In jedem Notizbuch steht kostenlos Rechenleistung und Speicher (begrenzter RAM, CPU, GPU, usw.) zur Verfügung, wird Colab für das Training der KI-Modell verwendet. Der MySQL Server, der die zentrale Schnittstelle im Datenaustausch darstellt, muss ebenfalls in der Analyse berücksichtigt werden. Der Datenbank-Server der Test-Umgebung und des Produktiv-System wird von Sylvenstein eingerichtet, der Datenbank-Server ist im Semi-Backend lokalisiert, und von einer Firewall geschützt, Zugriffe für die Colab-Umgebungen sind per SSL¹¹ verschlüsselt. Das bedeutet, um eine lauffähige Version des Projektes herzustellen, wird Rechenleistung für die KI-Systeme benötigt. Diese Rechenleistung soll von Colab stammen, um die Kosten des Projektes gering zu halten. Kosten für die Datenbank sind auch überschaubar, da bereits ein Datenbank-Server zur Verfügung steht, allerdings müssen trotzdem Kosten für die genutzte Rechen- und Speicherleistung der Datenbank berücksichtigt werden. Auf der anderen Seite lässt sich abschätzen, dass bei erfolgreicher Umsetzung des Projektes die Arbeitskraft eines Sekretärs, der für den Posteingang zuständig ist, teilweise entfallen würde. Damit allein würden deutliche firmeninterne Verwaltungskosten eingespart, was allein die Umsetzung dieses Projektes

¹⁰ Siehe Anhang Glossar: 8.4.6 Colaboratory

¹¹ SSL: https://de.wikipedia.org/wiki/Transport_Layer_Security Stand: 04.10.2021

rechtfertigen würde. Weitere Punkte werden nach Abschluss der Projekt-Planung detailliert in Punkt 3.7 (Kostenplanung) aufgelistet. Allerdings ist das Projekt, wie schon oben beschrieben, nicht nur rein wirtschaftlich zu betrachten, da es auch darum gehen soll Knowhow für zukünftige KI-Anwendungsfelder für Sylvenstein zu erschließen. Es folgen die geschätzten

Entwicklungskosten:

Personalkosten: $70h * 12,00€ = 840,00€$

Hard- & Software: 250,00€

Gesamt: 1090,00€

Um diese kurze Wirtschaftlichkeitsanalyse zusammenzufassen: die Entwicklungskosten des Projektes sind überschaubar und eine Amortisation ist realistisch.

3 Projektplanung

3.1 Erstellung des Lasten- und Pflichtenheftes

3.1.1 Lastenheft (Auszug)

Die Anwendung muss folgende Anforderungen erfüllen:

1. KI: Spam erkennen
2. KI: Tags aus Textinhalt erkennen
3. SEMI: Verknüpfung vom KI-System, Semi und Mail-Server

3.1.2 Pflichtenheft (Auszug)

Im folgenden Auszug aus dem Pflichtenheft wird die geplante Umsetzung der im Lastenheft gelisteten Anforderungen beschrieben:

- 1. Erstellen einer Schnittstelle für den Datenaustausch zwischen Semi, Mail-Server und KI-System**
 - a. MySQL – Datenbank:
 - i. Mitarbeiter-Zuständigkeit-Tabelle
 - ii. Test-System
 - iii. Produktiv-System
- 2. KI-System**
 - a. Laufzeit-Umgebung: Colab
 - b. Schnittstelle zur Datenbank: SQLAlchemy
 - c. Neuronales Netzwerk entwickeln:
 - i. Spam Erkennung
 - ii. Themen Erkennung
 - d. Iteratives Trainieren der Modelle
 - e. Export der Vorhersagen für Spam und Tag in Datenbank
- 3. Implementierung des KI-Systems**
 - a. Semi: Export von Emails in MySQL
 - b. KI: Vorhersagen treffen (Produktiv-System)
 - c. Semi: Verschieben der E-Mail (Produktiv-System)
 - i. Spam-Ordern
 - ii. Tags: Zum zuständigen Mitarbeiter

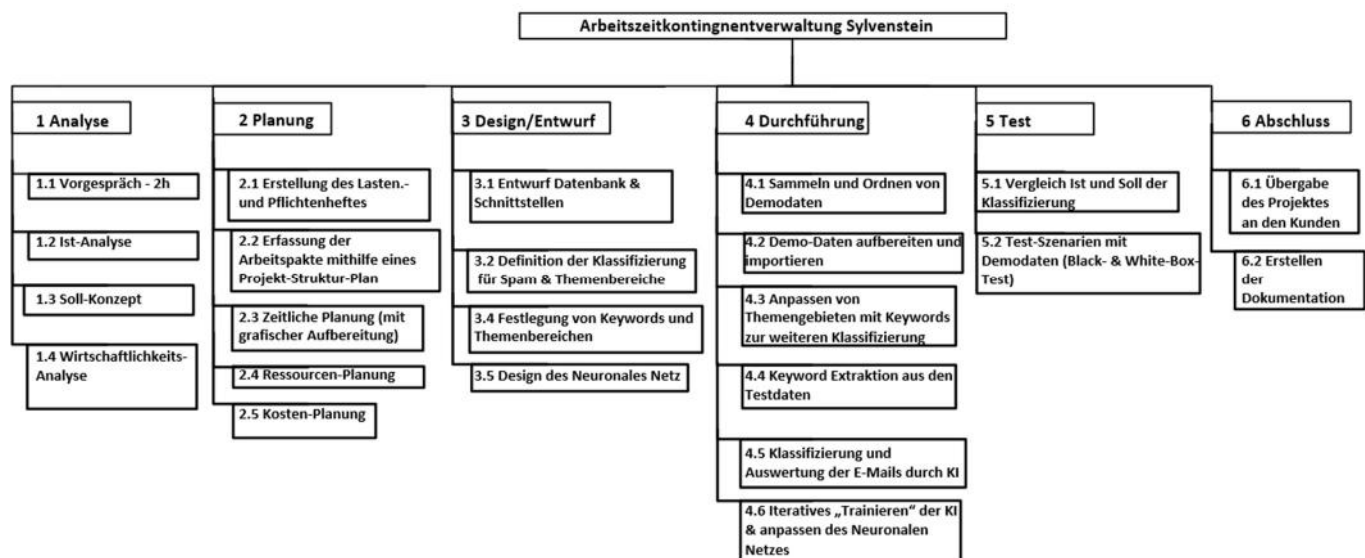
Abbildung 2- Auszug: Pflichtenheft

3.2 Erfassung der Arbeitspakete mit Zuordnung zu den Projektphasen

Arbeitspaketbeschreibung:		
Projekt: Klassifizierung von E-Mails mithilfe von KI	AP Nr.: 2.2	AP-Bezeichnung: Erfassung der Arbeitspakete mithilfe eines Projekt-Struktur-Plan
Beginn: 20.09.2021	Ende: 20.09.2021	Verantwortlich: Philipp Braun
Zu erzielende Ergebnisse: Alle Arbeitspakete sollen mithilfe eines PSP den Phasen zugeordnet werden		
Tätigkeiten: Auflistung der Arbeitspakete; Auflistung der Phasen; Zuordnung im Projektstrukturplan		
Voraussetzungen: Projektphase; Arbeitspakete;		
Unterschrift Projektleiter:		Unterschrift AP-Verantwortlicher:

Tabelle 2 – Beispielhafte Abbildung eines Arbeitspaketbeschreibung

Die Vorgangsliste enthält die logische und zeitliche Abfolge. Im Projektstrukturplan (PSP) werden dann auch die Phasen zugeordnet.



3.3 Projektphasen mit Zeitplanung

Phase	Aufgabe	Dauer-Phase	Dauer-Aufgabe
1. Analyse		6h	
	1.1 Vorgespräch		2h
	1.2 Ist-Analyse		1h
	1.3 Soll-Konzept		1h
	1.4 Wirtschaftlichkeitsanalyse		2h
2. Projektplanung		6h	
	2.1 Lasten- & Pflichten-Heft		1h
	2.2 Erfassung Arbeitspakete		1h
	2.3 Zeitliche Planung		1h

	2.4 Ressourcen-Planung		1h
	2.5 Kosten-Planung		2h
3. Design/Entwurf		4h	
	3.1 Entwurf Datenbank		1h
	3.2 Definition der Klassifizierung für Spam & Themen		1h
	3.3 Festlegung von Keywords und den dazugehörigen Themen		1h
	3.4 Design Neuronales Netzwerk		1h
4. Projektdurchführung		28h	
	4.1 Sammeln und Ordnen von Demodaten		6h
	4.2 Demo-Daten aufbereiten und Importieren		5h
	4.3 Themengebieten mit Keywords klassifizieren		3h
	4.4 Keyword Extraktion aus den Demo-Daten		6h
	4.5 Klassifizierung und Auswertung der E-Mail mithilfe von KI		2h
	4.6 Iteratives Training & anpassen des Neuronalen Netzes		6h
5. Testen		9h	
	5.1 Vergleich IST und Soll der Klassifizierung		4h
	5.2 Test-Szenarien mit Demodaten (Black- & White-Box-Test)		5h
6. Projektabschluss		17h	
	6.1 Übergabe des Projektes an den Kunden		1,5h
	6.2 Erstellen der Dokumentation		15,5h
Gesamt:		70h	70h

Tabelle - Vorgangsliste

3.4 Zeitliche Planung

Anschließend wird die Vorgangsliste für ein Gant-Diagramm genutzt, in dem der Projektablauf festgehalten wird. Dieser Ablauf dient zugleich auch als Planungs- und Kontrollinstrument für den Entwicklung.

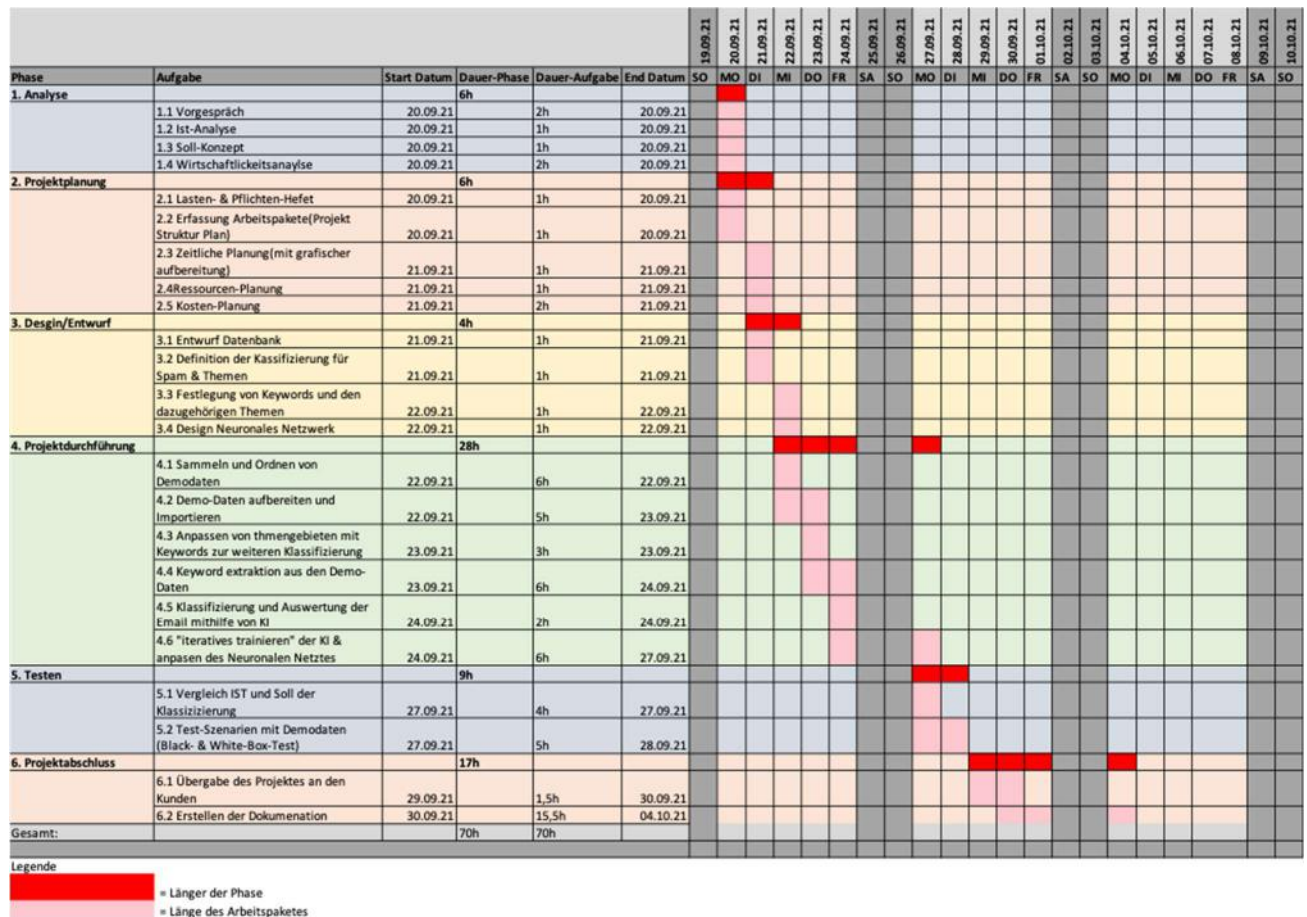


Abbildung 3 - Gantt Diagramm

3.5 Ressourcenplanung

Ressource	Bezeichnung	Funktion
Windows-PC	Arbeitsplatz	Arbeitsplatz
Colab.research	Colab	Rechenleistung bereitstellen
Open-Projekt	OpenProject	Projekt-Management
Github	Github	Versionskontrolle
Seminar-Verwaltungs-Software	Semi	Im- & Export von E-Mails
MySQL	MySQL	Zentrale Datenbank & Schnittstelle
Sylvenstein Mail-Server	Mail-Server	Trainingsdaten & Neu eintreffende E-Mails
Office 365	Office-Anwendungen	Erstellen der Dokumentation
Carbon.now.sh	Carbon	Erstellen von Quellcode-Abbildungen
Test-System	Test-System	Zum Trainieren der Modelle (Demodaten)
Produktiv-System	Produktiv-System	Implementierung/Projektabschluss
Mitarbeiter	Mitarbeiter	Festlegung der Themen & Tags

Tabelle 3 - Ressourcenplanung

3.6 Entwicklungsprozess

In diesem Projekt wurde auf einen agilen Entwicklungsprozess innerhalb eines Tickets-Systems gesetzt¹² (SCRUM¹³). Zuerst wird eine grobe Skizze der Grundfunktionen des Projektes entwickelt (Bare Bone). Diese lauffähige Grundversion des KI-System wurde von Anfang an in seiner

¹² Siehe: 3.4.1 Ressourcenplanung: OpenProject

¹³ Siehe Anhang Glossar: 8.4.1 SCRUM

Entwicklung Schritt für Schritt getestet. Diese Grundversion arbeitete mit Trainingsdaten und wird erst im zweiten Schritt an das Produktiv-System mit Daten aus der realen Welt angeschlossen. Im Test-System soll das iterative Training der KI-Modelle stattfinden. Zum Trainieren wird eine Feedbackschleife¹⁴ verwendet. Dieses Vorgehen bei der Softwareentwicklung entspricht ebenfalls dem agilen Vorgehen, weil das Trainieren der KI in Schleifen stattfindet. Erst wenn dieser Trainings-Zyklus abgeschlossen ist und die Gewünschte Genauigkeit bei den Vorhersagen erreicht wird, wird das Model mit echten E-Mails getestet werden.

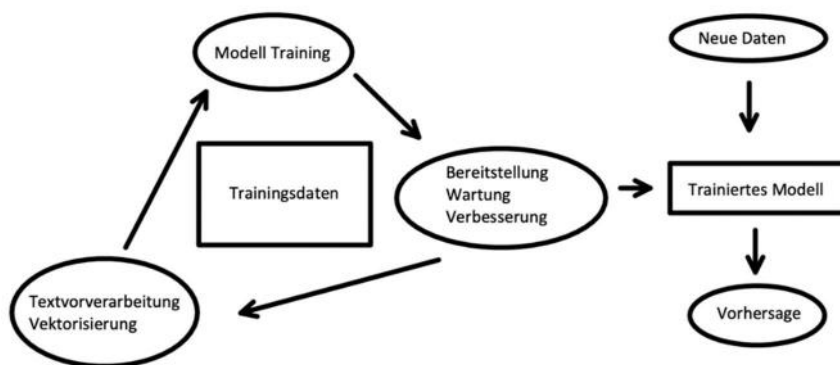


Abbildung 4 - Iteratives Trainieren

Bei Fehlfunktionen sowie weiteren Analyse-Feature-Anfragen wurden Tickets in Open Project erstellt und dem Auftragnehmer zugewiesen. Daraufhin wurden die Fehlfunktionen behoben, Features implementiert, und das Github- Repository aktualisiert. Auf diese Weise sind beispielsweise noch zusätzliche Features zur Keyword-Phrase-Extraktion (Schlüsselsatz Extraktion) entstanden. Die Dokumentation des Auftragnehmers soll auch künftig als Informationsquelle genutzt werden. Es wird eine Entwicklerdokumentation¹⁵ für Sylvenstein angefertigt.

3.7 Kosten-Planung

3.7.1 Make or buy Entscheidung

Aufgrund der Tatsache, dass in dem Bereich KI noch kein Know-How bei Sylvenstein vorhanden ist, lässt sich diese Frage leicht beantworten. Durch das eigenständige Programmieren der Software(Inhouse-Entwicklung) und die Anfertigung der Dokumentation soll gewährleistet werden, dass ein Wissenserwerb für zukünftige Anwendungsfälle stattfindet. Außerdem sind die Kosten für die Nutzung eines KI-Systems enorm und würden den Nutzen nicht rechtfertigen. Der größte Nutzen des Projektes liegt in der Einsparung von Arbeitsaufwand beim Sichten von E-Mails

¹⁴ Siehe: 5.6.1.2.3 Ablauf: Feedbackschleife

¹⁵ Siehe Anhang: 0Entwickler Dokumentation

für jeden einzelnen Mitarbeiter und in dem oben angesprochenem Wissenserwerb für zukünftige komplexere KI-Projekte.

3.7.2 Projektkosten

Eine firmeninterne Pauschale von 10,00€ pro Stunde wurde von der Geschäftsführung festgelegt. Diese Pauschale setzt sich zusammen aus: Stromkosten, Büromiete, Anschaffungskosten der Hard- und Software. Da die tatsächlichen Personalkosten aus Datenschutzgründen nicht herausgegeben werden dürfen, wird mit der festgelegten Pauschale kalkuliert. Der Stundensatz des Auftragnehmers wird auf 12,00 € festgesetzt: $12,00\text{€}/\text{h} \cdot 1680\text{h} = 20.160\text{€}$.

Die angenommene Stundenpauschale der Mitarbeiter liegt bei 25,00€, die Pauschale des Projektleiters wird auf 40,00€ festgelegt.

Ressource	Kosten (monatlich)
Datenbank-Server	5,00€
Colab.research	0,00€
MySQL	0,00€
Arbeitsplatz	10,00€
Gesamtkosten (mtl.)	15,00€

Tabelle 4 - Monatliche Kosten: Betriebsmittel

Phase	Mitarbeiter	Zeit	Kosten/h	Berechnung	Summe (mtl.)
Analyse	1 x Auszubildender	6,0h	12,00 €	$12,00\text{€} \cdot 6 = 72,00\text{€}$	72,00 €
	1 x Projektleiter	2,0h	40,00 €	$40,00\text{€} \cdot 2 = 80,00\text{€}$	80,00 €
Planung	1 x Auszubildender	6,0h	12,00 €	$12,00\text{€} \cdot 6 = 72,00\text{€}$	72,00 €
Design/Entwurf	1 x Auszubildender	4,0h	12,00 €	$12,00\text{€} \cdot 4 = 48,00\text{€}$	48,00 €
	1 x Mitarbeiter	1,0h	25,00 €	$25,00\text{€} \cdot 1 = 25,00\text{€}$	25,00 €
Durchführung	1 x Auszubildender	28,0h	12,00 €	$12,00\text{€} \cdot 28 = 336,00\text{€}$	336,00 €
Test	1 x Auszubildender	9,0h	12,00 €	$12,00\text{€} \cdot 9 = 108,00\text{€}$	108,00 €
	1 x Mitarbeiter	2,0h	25,00 €	$25,00\text{€} \cdot 2 = 50,00\text{€}$	50,00 €
Abnahme	1 x Auszubildender	15,5h	12,00 €	$12,00\text{€} \cdot 15,5 = 186,00\text{€}$	186,00 €
	1 x Projektleiter	1,5h	40,00 €	$40,00\text{€} \cdot 1,5 = 60,00\text{€}$	60,00 €
Summe:	-	-	-	-	+1.037,00 €
Betriebskosten mtl.	-	-	Ca. 0,0208€	Siehe Tabelle3	+15,00€
Gesamtkosten					1052,00€

Tabelle 5 - Berechnung Entwicklungskosten

3.7.3 Amortisierung

3.7.4 Kunde

Kunde des Projektes ist gleichzeitig auch der Auftraggeber (Christian Giebel), deshalb steht der optimale Einsatz und die Einsparungsmöglichkeiten von Entwicklerressourcen für Sylvenstein im Vordergrund. Sylvenstein ist ein kleines Unternehmen, aktuell sind alle Entwickler auch für den Support mit verantwortlich, das bedeutet, dass täglich Entwickler-Ressourcen zur Überwachung des Postfaches gebunden sind. Anstatt nun eine Teilzeitstelle für verwaltungstechnische Aufgaben zu schaffen, sollen diese Tätigkeiten teilweise automatisiert und von dem geplanten KI-System übernommen werden. Es wird also eine Teilzeitstelle in der Verwaltung und Entwickler-Ressourcen eingespart, diese Einsparungen sollen die Grundlage für die Amortisation dieses Projektes sein. Angenommen, die Teilzeitstelle eines Sekretärs mit 30 Wochenstunden würde

Sylvenstein 2.140,00 € monatlich kosten, dann hätte sich dieses Projekt nach einer Betriebsdauer von 2 Wochen amortisiert. Zweitens ist allerdings noch die Einsparung an Entwicklerressourcen bei der Amortisation zu berücksichtigen. Angenommen, fünf Mitarbeiter sparen täglich jeweils 5min Arbeitszeit, weil der tägliche Blick ins Postfach entfällt, und Mails vom KI-System automatisch auf Spam gefiltert und zum zuständigen Mitarbeiter verschoben werden. Berechnung der monatlichen Einsparung an Entwickler-Ressourcen:

$$5\text{min} \cdot 5 \text{ Mitarbeiter} \cdot 5 \text{ Tag} \cdot 4 \text{ Wochen} = 500\text{min}$$

$$500\text{min} \div 60\text{min} = 8 \frac{1}{3} h$$

$$8 \frac{1}{3} h \cdot 25,00\text{€/h} = 208 \frac{1}{3} \text{€}$$

208,33 € an messbarer Arbeitsleistung können beim Betrieb des KI-System monatlich eingespart werden. Zusammenfassend hat das Projekt wirtschaftliches Potenzial, die möglichen monatlichen Einsparungen überstiegen die Gesamten Entwicklungskosten bereits in der ersten Hälfte des ersten Betriebsmonats. Bei einem Entwicklungskosten von 1052,00€ kann sich Entwicklung des KI-Systems nach ca. 14 Tagen amortisieren:

Einsparung	Betrag Monatlich
Sekretär in Teilzeit	-2.140,00€
Entwickler-Ressourcen	-208,33€
Summe monatlich Einsparung	2348,33€

Tabelle - Berechnung Einsparung

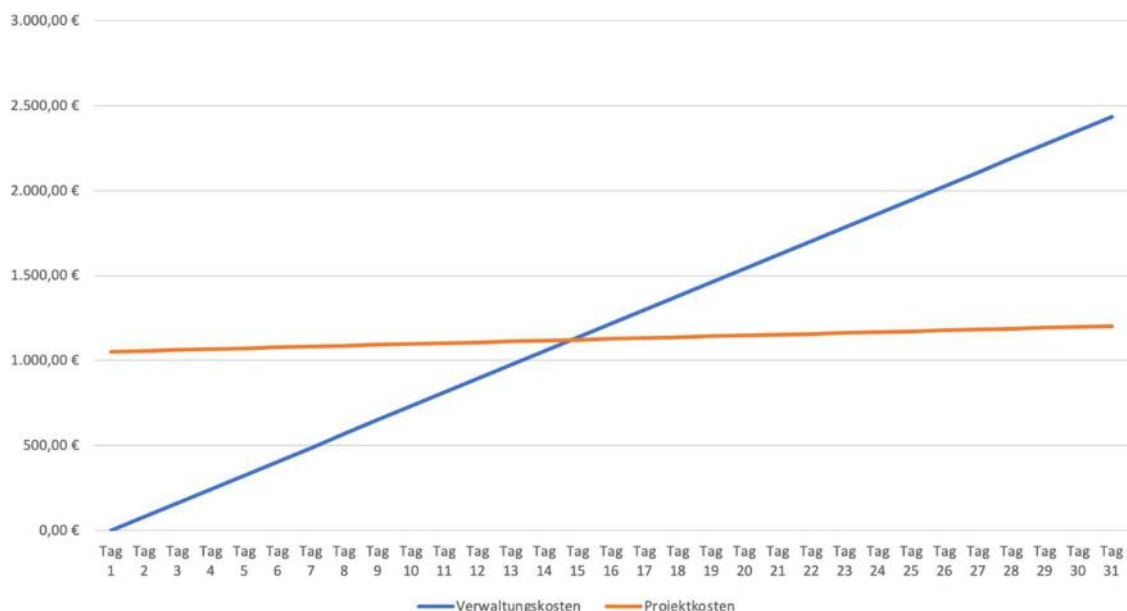


Abbildung 5 - Grafische Darstellung der Amortisierung

3.7.5 Abgrenzung: "reine Wirtschaftlichkeit"

Grundsätzlich ist zu sagen, dass dieses Projekt nicht nur rein wirtschaftlich betrachtet werden kann. Es geht bei diesem Projekt auch darum Knowhow zu erwerben, das in zukünftigen Projekten Anwendung findet. Außerdem soll bei der Programmierung ein modularer Ansatz verfolgt werden, die Textvorverarbeitung und die beiden KI-Systeme (Spam & Tags) werden unabhängig voneinander in verschiedenen Notebooks programmiert ausgeführt und trainiert.

3.7.6 Entwicklung

Dieser Entwicklungsprozess in Verbindung mit der zentralen Schnittstelle (MySQL) ermöglicht es, auch Daten und E-Mails aus anderen Quellen auszuwerten, zusätzlich lassen sich die KI-Modell via Objektserialisierung¹⁶ fertig trainiert in andere Projekte und Anwendungsbereiche überführen.

4 Design/Entwurfs-Phase

4.1 Entwurf Datenbank & Schnittstellen

4.1.1 Plattform: Colab

Die KI-Modelle sollen im Semi-backend implementiert werden und modular einsetzbar sein. Deshalb soll die Entwicklung der KI-Modelle in der Laufzeitumgebung von Colab stattfinden. Jedes KI-Modell wird in einem eigenen Notebook ausgeführt, um alle entwickelten Systeme unabhängig voneinander auch in zukünftigen Projekten einsetzen zu können. Dies ermöglicht die Entwicklung und das Trainieren der KI-Modelle in der Colab Laufzeitumgebung und den Export des Modells via Objektserialisierung in jede beliebige Python Anwendung.

4.1.2 Datenbank

Zentrale Schnittstelle dieses Projektes soll eine MySQL-Datenbank sein, die im Semi-Backend von Sylvenstein gehostet werden soll. Dort sollen zwei Datenbanken entstehen, eine für ein Test-System, in dem die Trainingsdaten abgelegt werden sollen und eine weitere für das Produktiv-System, in dem Semi neu eintreffende Emails ablegen soll. Das Test-System dient dazu, eine abgeschlossene Entwicklungsumgebung für das Trainieren der KI-Modelle zu schaffen. Das Produktiv-System ist dazu da, eine Implementierung der KI-Modelle vorzunehmen, wenn das Training erfolgreich abgeschlossen ist. Aufbau beider Datenbanken und ihrer Tabellen soll identisch sein, einziger Unterschied zwischen beiden Systemen ist, dass im Test-System mit Trainingsdaten und im Produktiv-System mit echten Daten gearbeitet wird. Diese Symmetrie

¹⁶ Siehe: Anhang Glossar: Objektserialisierung 8.4.7

beider Systeme soll die Implementierung des KI-Modells nach Abschluss der Testphase erleichtern. MySQL bildet die zentrale Schnittstelle zwischen Semi, dass vom Mail-Server Daten liefern soll. Auf der anderen Seite greifen die verschiedenen KI-Modelle auf die Datenbank zu, importieren die Emailldaten, analysieren sie und exportieren ihre Vorhersagen zurück in die Datenbank. Aufgrund der getroffenen Vorhersagen der KI-Modelle, soll Semi dann die Vorhersagen auslesen und die Emails im Mail-Server entsprechend verschieben. Hauptaufgabe der Datenbank ist es die Emailldaten für das Training und die Vorhersagen der KI bereitzustellen. Die Datenbank besteht aus zwei Tabellen, eine Tabelle ist zum Speichern der Emails, die andere zum Speichern der Themen und Tags. Im Anhang¹⁷ werden die Erstellungs-Skripte beider Tabellen aufgelistet, es ist zu beachten, dass extra der Datentyp Text gewählt wurde, da Text einen String mit einer Maximalen Länge von 65.535 Bytes aufnehmen kann. Dies ist nicht optimal für den Speicherbedarf der Datenbank, jedoch ist es notwendig, da in dieser Tabelle die rohen unverarbeiteten Emailldaten abgelegt werden soll. Um Fehler beim Import der Daten aufgrund Fehlenden Speichers zu vermeiden, ist Text notwendig. Außerdem werden im weiteren Verlauf dieses Projektes noch weitere Spalten hinzugefügt. Einige sind für das Labeln der Daten zuständig, andere um die Daten abzulegen, die noch dem Text-Preprocessing entstanden sind und wieder andere, um die Vorhersagen der KI aufzunehmen.

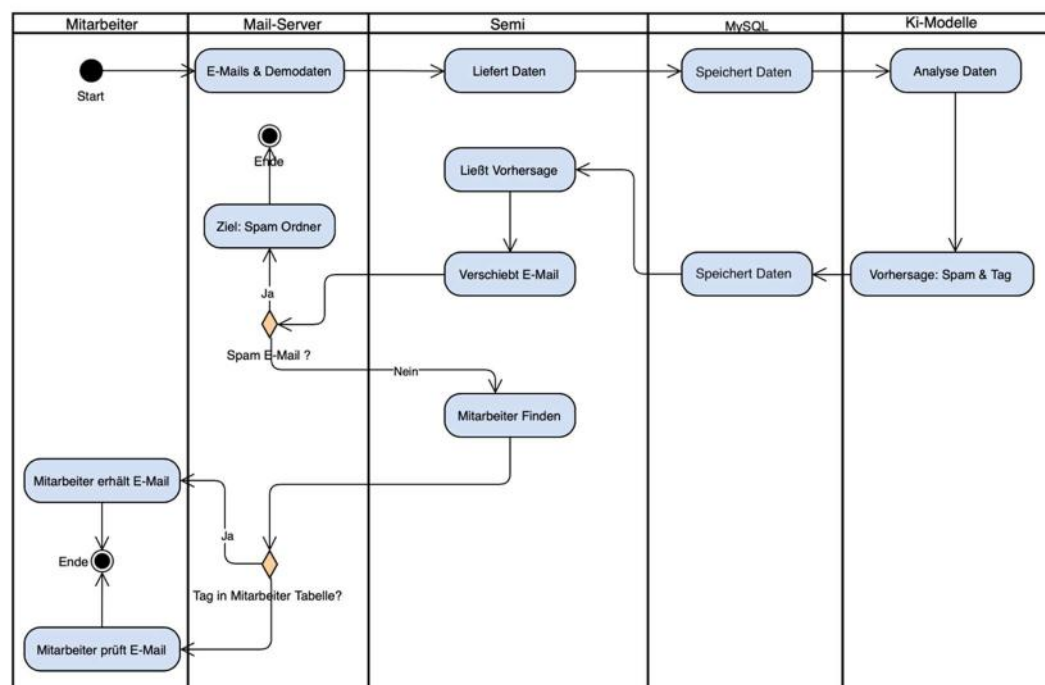


Abbildung 6 - UML Aktivitätsdiagramm: Grundkonzept

¹⁷ Siehe: 8.6.1.3 Erstellungs-Skripte für die Tabellen in der MySQL Datenbank

4.1.3 Programmiersprache und Bibliotheken

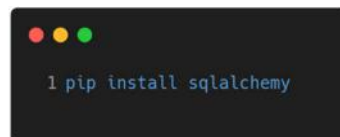
Es werden Python Bibliotheken für die Datenverarbeitung, Datenvisualisierung und den Austausch mit der MySQL-Datenbank benötigt, es steht vielfältige Open Source Software zur Verfügung. Im Anhang¹⁸ ist eine genaue Auflistung der Softwareressourcen zu finden sollen:

4.1.4 Schnittstellen

4.1.4.1 *technisch*

4.1.4.1.1 SQL-Alchemy

Zur Realisierung der Datenbank als Zentrale Schnittstelle musste eine Lösung gefunden werden um eine verschlüsselte Verbindung von Colab zur Datenbank herzustellen. Dafür wird die Open Source Bibliothek SQLAlchemy verwendet, die den Import und Export der Daten zwischen MySQL und Colab übernimmt. Zur Herstellung der Verbindung muss zuerst der entsprechende MySQL-Client im Colab Notebook installiert werden.



```
1 pip install sqlalchemy
```

Abbildung 7-Colab: install SQLAlchemy

Nach Import der Bibliothek im Notebook, kann die Verbindung mit der `create_engine()` Funktion erstellt werden. Diese Funktion benötigt den Korrekten Connection-String mit folgendem Aufbau:



```
1 engine = create_engine("dialect+driver://username:password@host:port/database")
```

Abbildung 8 -Colab: Connection String für SQLAlchemy

Innerhalb des Notebooks werden die Daten in einem Pandas Dataframe gespeichert und von dort aus auch verarbeitet. Deshalb werden nur zwei weitere Funktionen dieser Bibliothek in diesem Projekt verwendet, erstens der Import von Daten in ein Dataframe und der Export der verarbeiteten vom Dataframe zur MySQL-Datenbank.

Die benötigten Funktionen sind: `to_sql()` und `read_sql_table()`.

¹⁸ Siehe:

```
1 read_sql_table(tabellenName, ConnectionString)
2 engine = create_engine(ConnectionString)
3 dataframe = pandas.read_sql_table(tabellenName, engine)
4 return dataframe
```

Abbildung 9 - Funktion: Import von Daten mit SQLAlchemy

Die `read_sql_table()` Funktion ist in der Pandas Bibliothek enthalten, und benötigt nur eine Verbindung zur Datenbank diese wird mit dem `create_engine()` Befehl von SQLAlchemy aufgebaut.

```
1 To_sql(TabellenName, ConnectionString)
2 engine = create_engine(ConnectionString)
3 dataframe.to_sql(TabellenName, con=engine)
```

Abbildung 10 - Funktion: Export von Daten mit SQLAlchemy

Zum Herstellen einer verschlüsselten Verbindung zwischen Colab und MySQL muss erst die IP-Adresse des Notebooks in der Firewall des Semi Backendes freigeschaltet werden. Die Verbindung zwischen Colab und MySQL Datenbank findet verschlüsselt statt, als Verschlüsselungstechnik wird SSL19 verwendet.

4.1.4.1.2 Semi

Auf der anderen Seite soll Semi mit der MySQL Datenbank verbunden werden, um anhand der vorhergesagten Werte für Spam und Tag die Emails verschieben zu können. Semi soll zwischen Datenbank und Mail-Server stehen, um die Datensicherheit zu erhöhen. In Semi sind bereits Module zur Verwaltung des Mail-Servers enthalten. Im Test-System legt Semi die ausgewählten Demodaten in der E-Mail-Tabelle der Datenbank ab. Im Testsystem sollen keine Mails am Mail-Server verschoben werden. Im Produktivsystem soll Semi dann die eintreffende E-Mail im- und exportieren, die Vorhersagen der KI mit den Tags der Mitarbeiter vergleichen und bei Übereinstimmung verschieben. Allerdings ist zu erwähnen, dass dieses Semi-Modul schon existiert und nur vom Auftragnehmer die Verbindung zwischen Semi und MySQL hergestellt werden musste, um Semi zu implementieren. Außerdem ist noch zu erwähnen das aus dem Semi-Modul kein Quell-code gezeigt werden kann, da es sich um das geistige Eigentum von Sylvenstein handelt und nicht Bestandteil der Entwicklung des Autors ist.

4.1.4.1.3 Mail-Server

Das Verschieben der E-Mails wird von Semi übernommen und ist nicht Bestandteil dieser Dokumentation.

¹⁹ Siehe Anhang: Verbindungstest MySQL 8.6.1.2

4.1.4.1.4 Test-System

Im Test-System sollen Demodaten gesammelt und klassifiziert werden und das iterative Training der Neuronen Netze stattfinden.

4.1.4.1.5 Produktiv-System

Das Produktiv-System ist im Aufbau identisch mit dem Test-system, so kann gewährleistet werden, dass alle getesteten Funktion im Produktiv-System auch voll funktionsfähig sind.

4.1.4.2 Organisatorisch

4.1.4.2.1 Mitarbeiter

Die Mitarbeiter von Sylvenstein sind ebenfalls als Schnittstelle zu betrachten. Von ihnen ausgehend soll später eine Tabelle in der Datenbank mit folgendem Schema angelegt werden, um Firmeninterne Strukturen nicht öffentlich zu machen werden immer nur Beispielhafte Daten gezeigt:

Id	Vorname	Name	Abteilung	Thema	Tags
Int	Text	Text	Abteilung	Thema1, Thema2	Tag1, Tag2
Int	Text	Text	Abteilung	Kunde1, Thema3	Tag3, Tag4
Int	Text	Text	Abteilung	Technologie1	Tag5
Int	Text	Text	Abteilung	Kunde2,	Tag6
...

Abbildung 11 - MySQL: Aufbau der Mitarbeitertabelle

Es wurden persönliche Gespräche mit jedem Mitarbeiter geführt, um zu erfassen, für welche Abteilungen, Kunden und Technologien er zuständig ist. Es ist außerdem zu erwähnen, dass repräsentativ zu diesen erfassten Daten die Auswahl der E-Mails für die Trainingsdaten stattfand. In den Trainingsdaten müssen zu jedem Mitarbeiter und dessen Themen ausreichend Daten vorhanden sein. Alle ausgewählten Trainingsdaten werden unbearbeitet in der Tabelle für Emails abgelegt.

4.1.4.2.2 E-Mail-Verkehr

Der E-Mail-Verkehr an Sylvensteins allgemeines Postfach soll von der KI analysiert werden. Damit ist er auch indirekte Schnittstelle dieses Projektes. Der Allgemeine Eingang des Postfaches von Sylvenstein soll zukünftig die Input Daten im Produktiv-System liefern.

4.1.5 Textvorverarbeitung (Textpreprocessing)

4.1.5.1 Was ist Textvorverarbeitung?

Textpreprocessing (Textvorverarbeitung) ist die Verarbeitung natürlicher Sprache (Natural Language Processing, NLP). NLP stellt eine Überschneidung zwischen Linguistik, Informatik und KI da. Ziel ist die Verbindung zwischen Sprache und Computern, d.h., eine große Menge an Sprach Daten soll analysiert werden. Im Allgemeinen liegen die Daten, wenn sie analysiert werden sollen, immer in einem natürlichen Format von Sätzen oder Absätzen usw. vor. Bevor eine Analyse

vorgenommen werden kann, müssen die Daten umgewandelt und bereinigt werden, damit sie vom neuronalen Netz verarbeitet und verstanden werden können. Die Textvorverarbeitung ist ein zentraler Schritt für die Erstellung eines Modells für maschinelles Lernen. Nur mit gründlich vorverarbeiteten Daten sind die Vorhersagen des Modells zuverlässig. Grundsätzlich müssen bei der Textvorverarbeitung folgende Schritte stattfinden: Noise removal²⁰ und Kleinschreibung (kann in einigen Fällen aufgabenabhängig sein). Zusätzlich sollte die Textvorverarbeitung auch immer eine einfache Normalisierung²¹ umfassen. Alle weiteren Formen der Textvorverarbeitung sind Aufgaben abhängig. Zusammenfassend ist die Textvorverarbeitung der Prozess, mit dem der zu analysierende Text standardisiert wird. In den folgenden Abschnitten werden die Einzelschritte detailliert beschrieben, die zur bei der Textvorverarbeitung genutzt in diesem Projekt genutzt werden. Im agilen Entwicklungsprozess im Test-System werden nach jedem Schritt, die verarbeiteten Daten in einer neuen Spalte der Tabelle abgelegt und kontrolliert. Im Produktiv-System wird der gesamte Prozess der Textvorverarbeitung dann in einem Schritt ausgeführt.

4.1.5.2 *Textanreicherung*

Bei der Textanreicherung werden die ursprünglichen Textdaten mit Informationen ergänzt, die Sie zuvor nicht hatten. Das bedeutet, dass in der E-Mail-Tabelle eine neue Spalte erzeugt werden soll. In dieser Spalte werden Text der E-Mail, Betreff, Absender und Empfänger zur Analyse zusammengefasst werden.

4.1.5.3 *Noise removal(Geräuschentfernung)*

Bei der Geräuschentfernung geht es darum, Zeichen, Ziffern und Textteile zu entfernen, die bei der Textanalyse stören könnten. Geräuschentfernung ist stark vom Anwendungsfall abhängig. In Tweets können beispielsweise Sonderzeichen als Rauschen (Noise) vorkommen. Es gibt verschiedene Möglichkeiten, Geräusche zu entfernen. Dazu gehören z.B. die Entfernung von Satzzeichen, Sonderzeichen, Zahlen, HTML-Formatierung, anwendungsfallabhängige Schlüsselwörter (z.B. "WG" für Weitergeleitet), Quellcodeentfernung usw..

4.1.5.4 *Normalisierung*

Textnormalisierung ist der Prozess der Umwandlung von Text in eine kanonische²² (Standard) Form. Zum Beispiel wird das Wort "testen" und "testing" in "test", seine kanonische Form, umgewandelt. Die Textnormalisierung ist wichtig um Abkürzungen, Rechtschreibfehler usw. bei der Analyse berücksichtigen zu können.

²⁰ Siehe Abschnitt: 4.1.5.3 Noise Removal

²¹ Siehe Abschnitt: 4.1.5.4 Normalisierung

²² Quelle: <https://de.wikipedia.org/wiki/Normalform>

4.1.5.4.1 Rechtschreibung Korrigieren

Zum Korrigieren der Rechtschreibung soll in diesem Projekt die Python Bibliothek Textblob²³ eingesetzt werden. Textblob verwendet Statistiken über den Wortgebrauch in einer Sprache, um intelligente Vorschläge zu machen, welche Wörter korrigiert werden sollten. Um mit Textblob auch die deutsche Sprache Korrigieren zu können, muss eine statische Textdatei erzeugt werden. Die Datei sollte ausreichend Daten über die deutsche Sprache enthalten um Textblob ausreichend trainieren zu können²⁴. Der Folgende Code-Ausschnitt beschreibt wie diese Datei erzeugt werden kann:

```
1 from textblob.en import Spelling
2 import re
3
4 textToLower = ""
5
6 with open("germanSample.txt", "r") as f1:           # Open our source file
7     text = f1.read()                               # Read the file
8     textToLower = text.lower()                     # Lower all the capital letters
9
10 words = re.findall("[a-z]+", textToLower)          # Find all the words and place them into a list
11 oneString = " ".join(words)                       # Join them into one string
12
13 pathToFile = "train.txt"                          # The path we want to store our stats file at
14 spelling = Spelling(path = pathToFile)            # Connect the path to the Spelling object
15 spelling.train(oneString, pathToFile)              # Train
```

Tabelle 6 Textblob: Deutsche anhand eines Textes Lernen

4.1.5.4.2 Stemming(Stammformreduktion)

Stammformreduktion ist ein NLP-Prozess, der die Flexion (Veränderung)²⁵ in Wörtern auf ihre Stammformen reduziert. Flexion ist die Veränderung eines Wortes, um verschiedene grammatikalische Kategorien wie Zeitform, Kasus, Person, Zahl, Geschlecht und Stimmung auszudrücken. Ein Wort kann also in verschiedenen Formen existieren, aber diese Formen von Wörtern im selben Text führen zu Redundanz für NLP-Operationen. Daher wird die Stammformreduktion durchgeführt, wobei solche Stämme möglicherweise nicht einmal ein gültiges Wort sind. Beispiel dafür ist der Stamm von Reduktion, reduzieren und reduziertes: "redu", was an sich kein richtiges Wort ist.

4.1.5.4.3 Lemmatisierung

Lemmatisierung ist der Prozess, bei dem der Kontext verwendet wird, um ein Wort in seine sinnvolle Basis- oder Stammform umzuwandeln. In der Linguistik ist ein Lemma ein bedeutungsvolles Basiswort oder ein Wortstamm, der die Grundlage für andere Wörter bildet.

²³ Siehe Anhang: 8.4.4 Textblob

²⁴ Bei der Durchführung dieses Projektes wurde eine Textdatei mit langem deutschem Beispielttext erstellt(60.000Wörter)

²⁵ Quelle: Wikipedia <https://de.wikipedia.org/wiki/Flexion> Stand: 04.10.2021

Zum Beispiel ist das Lemma der Wörter "spielen" und "gespielt" "spielen". Der Unterschied zwischen Stemming und Lemmatisierung besteht darin, dass die Stammformreduktion zu ungültigen Wörtern führt, aber lemmatisierte Wörter führen immer zu sinnvollen Wörtern. Der Grund dafür ist, dass diese Wörter mit einem Wörterbuch abgeglichen werden.

4.1.5.4.4 Entfernung von Stoppwörtern

Stoppwörter sind eine Gruppe von Wörtern, die in einer Sprache häufig verwendet werden.

Beispiele für Stoppwörter im Englischen sind "a", "the", "is", "are" usw.. Die Intention hinter der Entfernung von Stoppwörtern ist, die wichtigen Wörter analysiert werden können, indem wir Wörter mit geringer Information aus dem Text entfernen. Wenn im Text beispielsweise "Wann ist das Banner fertig?" steht, sollte „ist“ und „das“ entfernt werden. Dies kann dadurch erreicht werden, dass alle Wörter aus einer Stoppwörterliste entfernt werden. Da deutsche und englische Texte verarbeitet werden sollen, müssen Stoppwörterlisten für Deutsch & Englisch verwendet werden.

4.1.5.4.5 Tokenization (Tokenisierung)

Ein Tokenizer zerlegt unstrukturierte Daten(Texte) in Informationsbrocken. Dadurch wird eine unstrukturierte Zeichenkette (Textdokument) sofort in eine numerische Datenstruktur umgewandelt. Die Token-Vorkommen in einem Dokument können direkt als Vektoren verwendet werden, die sich durch ein neuronales Netz analysieren lassen.

4.1.5.4.6 Textvektorisierung

Das Bag-of-Words-Modell²⁶ (BOW) ist eine Darstellung, bei der beliebiger Text in Vektoren fester Länge umgewandelt wird, indem gezählt wird, wie oft jedes Wort erscheint. Die BoW-Methode berücksichtigt nicht die Häufigkeit von Wörtern. Beispiel:

Text	der	Hund	sitzt	in	Hütte	mit
Der Hund sitzt	1	1	1	0	0	0
Der Hund sitzt in seiner Hütte	1	1	1	1	1	0
Der Hund mit Hütte	1	1	0	0	1	1

Tabelle 7 Erläuterung: Bag of Words

Ein anders Vorgehen zur Vektorisierung ist: Term frequency-inverse document frequency²⁷ (Td-if). Td-if berücksichtigt die Begriffshäufigkeit und die umgekehrte Dokumenthäufigkeit separat. Die Termfrequenz ist ein Maß für die Häufigkeit eines Wortes in einem Dokument. Die Termfrequenz für ein Wort ist das Verhältnis zwischen der Anzahl der Vorkommen eines Wortes in einem Dokument und der Gesamtzahl der Wörter in dem Dokument.²⁸ Die umgekehrte

²⁶ Quelle: https://en.wikipedia.org/wiki/Bag-of-words_model Stand 04.10.2021

²⁷ Quelle: <https://en.wikipedia.org/wiki/Tf-idf> Stand: 04.10.2021

²⁸ tf ('Wort') = Anzahl der Vorkommen von 'Wort' in einem Dokument / Gesamtzahl der Wörter in einem Dokument

Dokumenthäufigkeit²⁹ ist ein Maß für die Wichtigkeit eines Wortes. Sie misst, wie häufig ein bestimmtes Wort in allen Dokumenten des Korpus vorkommt.

4.2 Definition der Klassifizierung für Spam & Themenbereiche

4.2.1 Einleitung

In diesem Abschnitt soll der Prozess und die Notwendigkeit des Data-Labeling (Datenkennzeichnung), erläutert werden. KI ist nur so gut wie die Daten, mit denen sie trainiert wird. Qualität und Quantität der Trainingsdaten bestimmen den Erfolg des Projektes. Im Durchschnitt wird ca. 80% der Zeit, die für ein KI-Projekt aufgewendet wird, für Datenkennzeichnung³⁰, benötigt. Der Prozess der Datenkennzeichnung ist arbeitsintensiv, jede E-Mail aus dem Trainingsdaten-Set muss einzeln gekennzeichnet werden. Die Label(Kennzeichnungen) sind für das Training entscheidend, die Modelle werden vereinfacht gesagt, dadurch trainiert, dass das Model eine Vorhersage trifft, diese kann dann mit der erwarteten Kennzeichnung verglichen werden. Je nachdem ob die Vorhersage wahr oder falsch ist muss dann vor dem nächsten Trainingsdurchlauf (Epoch) das Neuronale Netz angepasst werden.

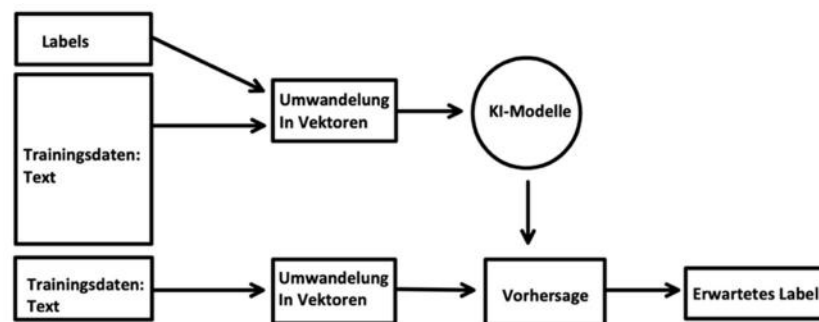


Abbildung 12 - Skizze: Bedeutung der Label der Trainingsdaten

4.2.2 Unlabeled Data(ungekennzeichnete Daten)

Die Trainingsdaten der E-Mails liegen in ungekennzeichneter Form im CSV³¹ Format vor und werden von Semi in der MySQL Datenbank abgelegt. Es wurde bei der Auswahl an Demodaten drauf geachtet, dass zu jedem Themengebiet, dass später zugeordnet werden soll ausreichend Trainingsdaten vorhanden sind. Um dies sicherzustellen werden die Daten vom Auftragnehmer gelabelt, randomisiert und vervielfältigt. Sollte die Genauigkeit bei der Auswertung nachträglich gesteigert werden, müssen entweder mehr Trainingsdaten gesammelt und gelabelt werden, oder das KI-Modell muss mehr Trainingsdurchläufe machen. Beide Möglichkeiten werden beim Iterativen Trainieren der KI geprüft.

²⁹ $\text{idf}(\text{'Wort'}) = \log(\text{Anzahl der Gesamtdokumente} / \text{Anzahl der Dokumente mit 'Wort' darin})$

³⁰ Labeln der Daten

³¹ Siehe Anhang Glossar: 8.4.3 CSV

4.2.3 Label Data (gekennzeichnete Daten)

Bei der Datenkennzeichnung muss in Absprache mit den Mitarbeitern darauf geachtet werden, dass das Kennzeichnen der Daten mit absoluter Präzision geschieht, jede falsche Kennzeichnung beeinflusst die Fehlerquote, der Vorhersagen der KI-Modelle.

4.3 Festlegung von Keywords und den dazugehörigen Themenbereichen

Der Auftragnehmer führt Gespräche mit jedem Mitarbeiter, um die Themengebiete, die Technologien und Kunden für die er zuständig ist zu erfassen. Auf Grundlage dieser Informationen wird die Mitarbeiter Tabelle in der MySQL Datenbank erstellt. In Absprache mit den Mitarbeitern wurden dann die Auswahl und die Label der Trainingsdaten repräsentativ ergänzt.

4.4 Design des neuronalen Netz

4.4.1 Supervised Machine Learning (überwachtes Lernen)

Beim überwachten Lernen wird ein Trainingsdatensatz verwendet, um das neuronale Netz so zu trainieren, dass es das gewünschte Ergebnis liefert. Dieser Trainingsdatensatz enthält gelabelte Daten, die es dem Modell ermöglichen, mit jeder Epoche zu lernen.

4.4.2 Neuronale Netze

Die Idee für den Entwurf eines neuronalen Netzwerkmodells ist eine Analogie zu der Art und Weise, wie biologische Organismen Informationen verarbeiten. Biologische Gehirne enthalten Neuronen, elektrisch aktivierte Nervenzellen, die durch Synapsen verbunden sind. Die so genannten künstlichen neuronalen Netze sind eine mathematische Funktion, die nach denselben Prinzipien entwickelt wurde. Es besteht aus elementaren Funktionen, den Neuronen, die in Schichten organisiert sind, die miteinander verbunden sind.

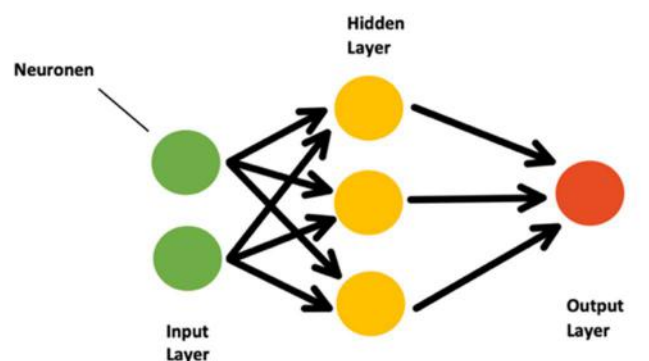


Abbildung 13 - Neuronales Netz: Vereinfachte Darstellung

Die Verbindungen in der grafischen Darstellung bedeuten, dass die Ausgabe einer Gruppe von Neuronen eine Schicht bilden. Die dann als Eingabe für die nächste Schicht von Neuronen dient. Dadurch wird eine Richtung definiert, in der Informationen von Schicht zu Schicht weitergegeben

werden, und wird daher als Feed-Forward-Netz bezeichnet. Im Allgemeinen verarbeitet ein neuronales Netz einfach einige Eingabedaten in die gewünschte Ausgabe. Ein Nachteil ist, dass neuronale Netze in der Regel von einer großen Anzahl von Parametern abhängen. Neuronale Netze, verarbeiten Trainingsdaten, indem sie die Interkonnektivität des menschlichen Gehirns durch Schichten von Knoten nachahmen. Jeder Knoten besteht aus Eingaben, Gewichten, einer Vorspannung (oder Schwelle) und einer Ausgabe. Wenn der Ausgangswert einen bestimmten Schwellenwert überschreitet, wird der Knoten "ausgelöst" oder aktiviert und leitet die Daten an die nächste Schicht im Netzwerk weiter. Neuronale Netze erlernen diese Zuordnungsfunktion durch überwachtes Lernen und passen sich auf der Grundlage der Verlustfunktion an. Wenn die Verlustfunktion bei oder nahe Null liegt, können wir sicher sein, dass das Modell die richtige Antwort liefert.

4.4.3 Design/Entwurf Neuronales Netzwerk

Beim Entwurf der Neuronalen Netze ist drauf zu achten, dass der Gesamte E-Mail-Text in beiden Modellen als Input für die Input-Layer dienen soll. Die Hidden-Layers werden in beiden Modellen erstmal gleich definiert, mit 2 Layer aus jeweils 256 Neuronen. In der Output-Layer liegt der Zentrale unterschied beider Modelle. Bei der Spam-Erkennung ist es wichtig zu beachten, dass nur ein einzelner Wert zurückgeben wird, nämlich 0 oder 1 für Spam oder nicht Spam-Mail. Bei der Vorhersage der Tags werden zwei Output-Neuronen benötigt, jedes Neuron kann einen oder keinen Wert für ein Tag zurückgeben. Damit kann dann ein, zwei oder kein Tag zurückgegeben werden. Die Anpassungen Hidden-Layer, der Algorithmen und Aktivierungsschwellenwerte innerhalb der Neuronalen Netzes sind Teil des iterativen Trainings der Modelle. Nach jeder Iteration sollen die Parameter angepasst werden, bis die Vorhersagen der Modelle die gewünschte Genauigkeit erreicht haben.

5 Projekt Durchführung

5.1 Sammeln und Ordnen von Demodaten

5.1.1 Export von Emails

Ausgangspunkt dieses Projektes sind die Trainingsdaten, die für die KI-Modelle in ausreichender Form vorhanden sein müssen. Das Beschaffen und die Textvorverarbeitung dieser Trainingsdaten stellt damit den ersten Schritt der Projektdurchführung da.

5.1.2 Outlook

Die E-Mails treffen auf einem Outlook-Mail-Server ein, dieser Mail-Server ist mit Semi verknüpft, so dass ein Export von E-Mail-Daten im CSV-Format möglich ist. Es wird bei diesem Export eine repräsentative Auswahl für die Trainingsdaten getroffen.

5.1.3 CSV

Sobald die Daten im CSV-Format vorliegen, steht dem weiteren Aufbereitungsprozess nichts mehr im Weg. Das CSV Format kann mithilfe der Pandas Bibliothek in ein Dataframe eingelesen werden und dann mithilfe von SQLAlchemy in der MySQL Datenbank abgelegt werden.

5.1.4 Test-System

In der Testumgebung sollen verschiedene Python Bibliotheken zur Textvorverarbeitung und der Entwicklung der KI-Modelle importiert und ausgeführt werden. Die Daten werden mithilfe von SQLAlchemy im- und exportiert. Hier wird dann der Trainingsprozess der KI-Modelle stattfinden.

5.1.5 Colab

Colab ist eine Laufzeitumgebung im Notebookformat in der einzelne Code- oder Textzellen verfasst und ausgeführt werden können. Dieses System eignet sich sehr gut zur Programmierung und Dokumentation des Quellcodes. Das Notebookformat erleichtert den agilen Entwicklungsprozess. Die Versionierung des Entwicklungsprozesses der Notizbücher wird in einem privaten Github Repository von Sylvenstein vorgenommen.

Zur Dokumentation werden vier Notizbücher erstellt:

1. Data-Import & Export: MySQL-Server-Connection mit SQLAlchemy
2. Textvorverarbeitung (Textpreprocessing)
3. Entwicklung & Training: KI-Modell – Vorhersage: Spam
4. Entwicklung & Training: KI-Modell – Vorhersage: Tags

5.2 Trainingsdaten aufbereiten und importieren

5.2.1 MySQL

Zur Speicherung der Daten wird eine MySQL Datenbank aufgesetzt. Die Datenbank-Server dazu werden von Sylvenstein bereitgestellt. Es wird eine Datenbank verwendet, damit erstens die Datensicherheit gewährleistet ist, zweitens um die Vorhersagen des Modells speichern, auswerten und verbessern zu können. Und drittens wird die Datenbank als zentrale Schnittstelle zwischen

Semi und den KI-Modellen fungieren. Die Datenbank und ihre Tabellen werden wie in der Design/Entwurfs-Phase beschrieben, implementiert³².

5.2.2 Datenkennzeichnung (Data labeling)

Um im Zeitplan dieser Projektarbeit zu bleiben, wir einem ausgewählten Datensatz von 150 E-Mails aus allen Themenbereichen begonnen. Die Daten werden gekennzeichnet und mit Text angereichert, daraufhin randomisiert und vervielfältigt, sodass am Ende 1500 gekennzeichnete E-Mails als Trainingsdatensatz zur Verfügung stehen. Diese Datenmenge sollten mit ausreichender Anzahl an Trainingsdurchläufen (Epoch) zu aussagekräftigen Vorhersagen führen. Sollte die gewünschte Genauigkeit nicht erreicht werden können ist zu beachten, dass dann einfach nur mehr Daten dem Trainingsdatensatz hinzugefügt werden müssen, um die Genauigkeit zu erhöhen.

5.2.2.1 Spam

Das Kennzeichnen der Daten erfolgt E-Mail für E-Mail. Den Trainingsdaten wird eine Spalte namens Spam hinzugefügt. Diese soll 1 oder 0 für Spam oder nicht Spam enthalten. Um eine ausreichende Qualität beim Setzen der Tags zu haben ist es an dieser Stelle wichtig, dass die Kennzeichnung der Daten vom Auftragnehmer und nicht automatisiert vorgenommen wird.

5.2.2.2 Tags

Bei den Tags soll ein oder zwei String Werte als Tag der E-Mail in einer Liste eingetragen werden und als Spalte namens Tags hinzugefügt werden. Das Kennzeichnen der Daten erfolgt E-Mail für E-Mail. Hierbei wird Betreff, Inhalt, Absender und Empfänger betrachtet und versucht mindestens ein passendes Tag für das Thema, den Kunden oder die Technologie zu finden. Aus Datenschutzgründen kann an dieser Stelle nur eine beispielhafte Darstellung gezeigt werden.

Id	Betreff	Inhalt	Absender	Empfänger	...	Spam	Tags
1	Beispiel Betreff	Lorem ipsum dolor sit amet ...	Kunde1	Autor1	...	0	Kunde1, Website1
2

5.3 Anpassen von Themengebieten mit Schlüsselworten zur weiteren Klassifizierung

5.3.1 Klassifizierung der Themengebiete

Um die Kennzeichnung der Daten zu überprüfen, wird eine Schlüsselwort Extraktion durchgeführt. Da sich die extrahierten Schlüsselwörter mit den Kennzeichen der E-Mail decken, kann die Klassifizierung der E-Mails als erfolgreich angesehen werden.

³² Siehe Anhang: 8.6.1.3 Erstellungs-Skripte für die Tabellen in der MySQL Datenbank

5.3.1.1 Mitarbeiter


In Absprache mit dem zuständigen Mitarbeiter, wird der Datensatz nur noch um E-Mails für fehlenden Themen, ergänzt. Dazu werden Einzelgespräche mit den Mitarbeitern geführt, in denen die Mitarbeiter die Klassifizierung/bzw. die Kennzeichnung ihrer E-Mails beurteilen. Bei fehlerhafter Klassifizierung wurden zusammen aussagekräftige E-Mails für dieses Thema ausgewählt, gekennzeichnet und ergänzt.

5.3.1.2 MySQL

Nachdem nun die Trainingsdaten repräsentativ dem realen E-Mail-Verkehr abbilden, sollen die Themen der Mitarbeiter in der MySQL Datenbank in der Mitarbeiter Tabelle abgelegt werden. Die Speicherung der Mitarbeiter und E-Mail-Daten erlaubt es nachträglich noch weitere Themen und die zugehörigen Daten einfach zu ergänzen, so wird auch hier ein modular skalierbarer Ansatz für die Lernfähigkeit der KI-Modell verfolgt.

5.4 Textvorverarbeitung & Schlüsselwort Extraktion

In diesem Abschnitt näher auf den Quellcode und die verwendeten Bibliotheken eingegangen werden. Alle Schritte sind für die Textvorverarbeitung notwendig, aber bspw. mit welcher Abfolge der Einzelschritte die besten Ergebnisse erzielt werden können, wird sich erst bei den ersten Vorhersagen der KI-Modelle zeigen. Das bedeutet, dass es möglich ist, dass die Textvorverarbeitung im Laufe des Entwicklungsprozesses nochmal angepasst werden muss. Deshalb werden alle Schritte modular entwickelt und abschließend zu einer Funktion zusammengesetzt, die dann auf jede Zelle des Dataframes angewendet werden kann. Es folgt der zugehörige Quellcode:



```
dataframe['Analysis'].apply(function_textpreprocessing)
```

Abbildung 14 - Quellcode: Textvorverarbeitung auf Dataframe anwenden

5.4.1 Textanreicherung

Die Verarbeitung der Daten findet in Colab statt, dort wird die Bibliothek SQLAlchemy verwendet um mit den aus MySQL importierten Daten umzugehen. Die Daten werden mithilfe der `read_from_MySQL()`³³ Methoden ausgelesen und anschließend zur Weiterverarbeitung in einem Pandas Dataframe abgelegt. Zur Textanreicherung sollen nun die Spalten: Betreff, Text, Absender,

³³ Siehe: 4.1.4.1.1 SQL-Alchemy

Empfänger und Tags zur Analyse Spalte zusammengefasst werden. Es folgt der dazu nötige Quellcode:

```
1 # Combining Columns for single input of analysis
2 dataframe["Analysis"] = dataframe["Betreff"]
3   + dataframe["Text"]
4   + dataframe["Absender"]
5   + dataframe["Empfänger"]
6   + dataframe["Tags"]
```

Abbildung 15 Quellcode: Textanreicherung

5.4.2 Geräuschentfernung (Noise removal)

In diesem Projekt macht es Sinn, Satzzeichen und E-Mail spezifische Tags zu entfernen. Zur Geräuschentfernung muss der Text als unformatierter String betrachtet werden. Dazu eignet sich die RegEx-Bibliothek. Diese lässt sich mithilfe von: "import re" in Colab einbinden. Beim Umgang mit regulären Ausdrücken wird Raw Strings als r'expression' verwendet.

```
1 # def function for removing noise
2 def removeNoise(text):
3     re.sub(r'[\.\?!\,\:\;\'\\"REWG]', '', text)
4
5 # Apply function
6 dataframe['Analysis'].apply(removeNoise)
```

Abbildung 16 - Quellcode: Geräuschentfernung

5.4.3 Normalisierung

5.4.3.1 Rechtschreibung Korrigieren

Python bietet viele Module für diesen Zweck, die das Schreiben einer einfachen Rechtschreibprüfung übernehmen können. Eine dieser Bibliotheken ist TextBlob, die für die Verarbeitung natürlicher Sprache verwendet wird und eine intuitive API bietet. Zunächst muss TextBlob installiert werden. Quellcode für die Installation in Colab:

```
1 pip install -U textblob
```

Abbildung 17 Quellcode: Colab Installation Textblob

TextBlob baut auf NLTK auf und wird daher ebenfalls mit der Installation mitgeliefert.

Die Anwendung der correct()-Funktion. Die detect()-Funktion ist ebenfalls Teil von TextBlob und liefert als Ergebnis die Sprache des Textes als String zurück, bspw. "de" oder "en" usw. Es folgt ein Quellcodeauszug, der die Sprache des Textes erkennt, und passend in Deutsch oder Englisch die

Rechtschreibung korrigiert, die Implementierung der deutschen Sprache wurde in der Entwurfsphase beschrieben:

```
def clean_blob_correct(text):  
    if detect(text) == 'en':  
        blob = TextBlob(text)  
    elif detect(text) == 'de':  
        blob = TextBlobDE(text)  
    return blob.correct().split()
```

Abbildung 18 - Quellcode: Rechtschreibkorrektur mit Textblob

5.4.3.2 Stemming

Verwendet wird in diesem Projekt der Snowballstemmer von NLTK diesem muss mit: "pip install snowballstemmer" im Colab Notizbuch installiert werden. Es Folgt der Quelle-Code der Stemming-Funktion.

```
1 # Imports  
2 from nltk.stem import SnowballStemmer  
3  
4 # def stemmer  
5 snowball = SnowballStemmer(language='english')  
6  
7 # def stemming function  
8 def snowStemmer(text):  
9     stemmed_text = ' '.join([snowball.stem(word) for word in text.split()])  
10    return stemmed_text.split()
```

Abbildung 19 - Quellcode: Stammformreduktion

5.4.3.3 Lemmatisierung

Bei der Lemmatisierung wird ein sogenanntes Wörterbuch verwendet. In diesem Projekt wird NLTK zum NLP eingesetzt, deshalb wird die Lemmatisierung mithilfe von WordNet durchgeführt. Die Definierte Funktion kann ebenfalls mithilfe von: ".apply(lemmatize_text)" auf die gewünschte Spalte des Dataframes angewendet werden.

```
1 # Imports  
2 from nltk.stem import WordNetLemmatizer  
3  
4 # def lemmatizer  
5 lemmatizer = WordNetLemmatizer()  
6  
7 # def stemming function  
8 def lemmatize(text):  
9     lemmatized_text = " ".join([lemmatizer.lemmatize(word) for word in text.split()])  
10    return lemmatized_text.split()
```

Abbildung 20 - Quellcode: Lemmatisierung

5.4.3.4 Entfernung von Stoppwörtern

Da in diesem Projekt Texte in deutsche und englische Sprache verarbeitet werden sollen, ist darauf zu achten, dass die Stopword-Liste um Deutsche bzw. Englische Stopword ergänzt wird. Es folgt der Quellcode, der die Stopwortlisten ergänzt, und die Funktion zur Anwendung auf das Dataframe wird definiert:

```
# use German stop word set
stop = set(stopwords.words('german'))

# add extra example stopwords which are not in list
stop.update({'der', 'die', 'das', 'dass', 'koennen', 'sind', 'the'})

# add english stopwords to stopword-list
stop.update(set(stopwords.words('english')))

# define funktion for stop-word-removal, which can be applied to df['col']
def clean_stop(text):
    stop_free = ' '.join([word for word in text.lower().split() if word not in stop])
    return stop_free.split()
```

Abbildung - Quellcode: Stoppwortlisten ergänzen & Funktionsdefinition

5.4.3.5 Schlüsselwort Extraktion (Keyword Extraction)

Aus den Texten werden fünf Schlüsselwörter und Schlüsselsätze extrahiert und jeweils in einer extra Spalte des Dataframes abgelegt. Es folgt der Quellcode, um mit der Gensim-Bibliothek fünf Schlüsselwörter zu extrahieren:

```
# Import
from gensim.summarization.summarizer import summarize
from gensim.summarization import keywords

# Intializing Counter and New Col in DF
df['5_extracted_keywords'] = NULL
i = 0

# While-Loop
# Extract Keywords for each row in DF
# And save extracted Keyword in new DF['5_extracted_keywords']
while i <= len(df):
    df['5_extracted_keywords'][i] = keywords(df['Analysis'][i], words=5)
    i += 1
```

Abbildung 21 - Quellcode: Schlüsselwort Extraktion

Der nächste Quellcode beschreibt den Extraktionsvorgang von fünf Schlüsselsätzen:

```
# imports & Downloads
import nltk
from nltk import word_tokenize, sent_tokenize
nltk.download('punkt')
nltk.download('stopwords')

# Initializing Counter and New Col in DF
df['5_extracted_keywords_Phrases']=0
rake_nltk_var = Rake()
i = 0

# While-Loop
# Extract Keyword-Phrases for each row in DF
# And save extracted Keyword-Phrase in new DF['5_extracted_keywords_Phrases']
while i <= len(df):
    rake_nltk_var.extract_keywords_from_text(str(df['Text'])[i]))
    df['5_extracted_keywords_Phrases'][i] = rake_nltk_var.get_ranked_phrases()[:5]
    i += 1
```

Abbildung 22 - Quellcode: Schlüsselsätze extrahieren

5.4.4 Abgrenzung: Projektantrag

Zu diesem Zeitpunkt der Projektdurchführung wurde festgestellt, dass die Extraktion der Schlüsselwörter und Sätzen nicht zielführen für die Vorhersagen der KI-Modelle ist. Die Extraktion macht Sinn, um den Inhalt beispielsweise die Gefühle bei Kundenbewertungen festzustellen, hat aber keinen direkten Einfluss auf die Vorhersagen, solange die Textvorverarbeitung ausreichend stattgefunden hat.

5.4.4.1 Ausblick: Topic Modelling

Um den zeitlichen Rahmen dieses Projekts nicht zu gefährden, soll an dieser Stelle nur auf den weiteren Nutzen eingegangen werden, den Topic Modelling bietet. Mithilfe von Topic Modelling kann der E-Mail-Datensatz in Cluster nach Themen geordnet eingeteilt werden. Topic Modelling könnte eingesetzt werden um, die Mitarbeiter Tabelle und ihre Themen zu Aktualisieren. Bei der Themenmodellierung handelt es sich um einen unbeaufsichtigten Ansatz zur Erkennung oder Extraktion von Themen durch die Erkennung von Mustern, die die Daten in verschiedene Teile unterteilen. Dies geschieht durch die Extraktion der Muster von Wortclustern und der Häufigkeit von Wörtern im Dokument.

5.5 Klassifizierung und Auswertung der E-Mails durch KI

5.5.1 Implementierung der KI-System

5.5.1.1 Definition des Neuronalen Netzes

Ausgehend von den Planungen in dieser Phase, sind in diesem Projekt zwei leicht unterschiedliche Modelle nötig. Für die Spamerkennung wird nur eine Output-Neurone benötigt, für die Tags zwei. Die Ausgabe der Spam Erkennung wird als booleschen Wert für Spam oder nicht Spam definiert.

Bei dem Erkennen der Tags ist der Output als BoW³⁴ definiert. Deshalb soll in diesem Fall ein Array, von der Länge n*Tag, zurückgegeben werden, dieses Array enthält dann eine Liste von booleschen-werten, die den Tags entsprechen. Je nach dem an welcher Stelle das neuronale Netz die booleschen Werte auf 1 gesetzt hat.

Vorhersage:	Array([[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
Tags (Bag of Words) :	Array(['Ampelsystem', 'Apps', 'Banner', 'Bestellung', 'CitrixFehler', 'Dokumentation', 'Fake', 'Fraud', 'Gehe', 'Kalender', 'Mailserver', 'Marketing', 'Meeting', 'Mockups', 'RecuCare', 'Reporting', 'SQL', 'Salzgitter', 'Sammelbuchung', 'Semi', 'Seminare', 'Seminarkonfigurator', 'Spam', 'Sylvenstein', 'TestKunde', 'WebsiteFehler', 'WidgetVeranstaltungen', 'Zertifikat', 'elearning', 'pdfUpload'], dtype=object)

Tabelle 8 - Vorhersage/Tags

Mit diesem Befehl kann dem Array der richtige wert zugeordnet werden:

```

1 In: print(multilabel.inverse_transform(
2       Array([[0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]])
3     )
4   )
5
6 Out: [('Gehe', 'Reporting')]
```

Abbildung 23 – Quellcode: BoG Zuordnung

Es folgen zwei kurze Codeausschnitte, die den Aufbau der beiden Modelle darstellen.

```

1 # Modell container
2 model = tf.keras.models.Sequential()
3
4 # Input-Layer
5 model.add(tf.keras.layers.Dense(256, input_shape=x_train.shape, activation='sigmoid'))
6 # Hidden-Layer
7 model.add(tf.keras.layers.Dense(256, activation='sigmoid'))
8 # Output-Layer
9 model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Abbildung 24 Quellcode: Modell Spam

```

1 # Modell container
2 model = tf.keras.models.Sequential()
3
4 # Input-Layer
5 model.add(tf.keras.layers.Dense(256, input_shape=x_train.shape, activation='sigmoid'))
6 # Hidden-Layer
7 model.add(tf.keras.layers.Dense(256, activation='sigmoid'))
8 # Output-Layer
9 model.add(tf.keras.layers.Dense(1, output_mode="multi_hot", activation="sigmoid"))
```

Abbildung 25 – Quellcode: Modell Tags

³⁴ Siehe: Textvektorisierung – Bag of Words

5.5.1.2 Erklärung

Um die Modelle zu erzeugen, wird mit „Sequential()“ ein Container erstellt, in den sequenziell die Ebenen des Modells mit „add()“ angefügt werden. Dabei wird mit „name“ der Name des Layers, mit „unit“ die Anzahl der Neuronen und mit „activation“ die Aktivierungsfunktion festgelegt. Die Eingabeform wird mit `input_shape=x_train.shape` auf das Format der Daten angepasst. Die Aktivierungsfunktion ist in beiden Fällen eine Sigmoid-Funktion³⁵. Das bedeutet vereinfacht gesagt, dass der Wert, der zurückgegeben wird entweder 0 oder 1 ist, je nach dem, was dem Wert näher ist. Für einen Wert von 0,3 würde es bedeuten, dass 0 zurückgegeben wird, für 0,6 würde 1 ausgegeben werden. Im Fall der Spam-Erkennung reicht es aus, 1 Neuron mit diesem Wert zurückzugeben. Bei der Erkennung von Tags wird wie oben beschrieben ein Array mit booleschen Werten der Länge „n*Anzahl Tags“ zurückgegeben, dies wird durch `output_mode='multi_hot'` erreicht.

5.5.1.3 Schnittstellen

Da bisher nur die Funktionsweise der KI-Modelle beschrieben wurde, soll in diesem Abschnitt der Ablauf, die Übergabe und die Verarbeitung der Daten im Vordergrund stehen.

5.5.1.3.1 SQLAlchemy

Ausgehend von der Datenbank können die Daten mit der Bibliothek SQLAlchemy in die Colab Notebooks importiert werden.

5.5.1.3.2 Dataframe

In Colab werden die Daten in einem Pandas Dataframe abgelegt, und von dort vom KI-Modell verarbeitet. Die Verarbeitung umfasst die Textvorverarbeitung, das Trainieren der KI-Modelle im Test-System und die Vorhersagen im Produktivsystem. Im Test-System werden die Vorhersagen für Spam und Tags ausgewertet, bis die gewünschte Genauigkeit erreicht ist. Im Produktivsystem werden die KI-Modelle nicht mehr trainiert, sondern es werden nur noch die Vorhersagen zu Spam und Tag bei jeder neu eintreffenden E-Mail gemacht. Nach Verarbeitung der Daten werden die Werte für Spam und Tags in zwei neuen Spalten des Dataframes abgelegt. Der Export der Daten aus dem Notizbuch wird wieder mithilfe von SQLAlchemy bewerkstelligt.

5.5.1.3.3 MySQL

Im Testsystem werden die Daten dann in einer neuen Tabelle („emails_processed“) in MySQL abgelegt, um den Verarbeitungsprozess der Daten zu visualisieren. Im Produktivsystem werden nur die Vorhersagen der KI-Modelle der E-Mail-Tabelle hinzugefügt.

³⁵ Siehe Anhang Glossar: 8.4.2 Sigmoid-Funktion

5.5.1.3.4 Semi

Im Testsystem findet noch kein Verschieben der E-Mail statt, es wird nur eine Auswertung der Vorhersagen vorgenommen, aufgrund dessen dann wieder das KI-Modell angepasst wird. Im Produktivsystem liest Semi automatisch die Vorhersagen zu jeder E-Mail und verschiebt diese dann bei Übereinstimmung in das Postfach des zuständigen Mitarbeiters oder den Spam Ordner.

5.5.1.3.5 Emails - Mailserver

Das Verschieben der E-Mail wird von einem Semi-Modul übernommen, dies hat den Vorteil, dass KI-Modelle und vom restlichen System getrennt bleiben, dies erhöht die Datensicherheit.

5.6 Iteratives Training der KI-Modelle und Anpassung des neuronalen Netzes

5.6.1 Trainieren

5.6.1.1 Erklärung

Mit `compile()` werden der Optimierer (Optimizer), die Loss Funktion und zusätzliche Metriken konfiguriert³⁶. Mit `fit()` werden dem Modell die Input- und Zieldaten für das Training übergeben und mit „epochs“ die Anzahl der Iterationen festgelegt.



```
# Modell konfigurieren
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

# Modell trainieren
model.fit(x = input, y = target, epochs = 500)
```

Abbildung 26 Quellcode: KI-Modell kompillieren

Neuronale Netze arbeiten mit Trainingsdaten, bei denen der Output bekannt ist (gekennzeichnete Daten). Zum Trainieren des Modells wird die Vorhersage mit den gekennzeichneten Daten verglichen. Aus diesem Vorgang heraus ("Trial-and-Error") passt sich das neuronale Netz im Verlauf des Training selbständig an. In unserem obigen Quellcode Ausschnitt bekommt x den Input übergeben, die Input Variabel enthält die Daten, denen das Label entfernt wurde. Y bekommt die Target-Variabel übergeben, diese enthält das passende Label. Der Wert der an epochs übergeben wird bestimmt die Anzahl der Iterationen. Nachdem das Modell die Input Daten verarbeitet hat und eine Vorhersage getroffen hat, wird die Vorhersagen des Modells über eine Loss Funktion mit dem Target verglichen. Die Verbesserung des Modells geschieht über die Gewichte, mit denen die Aktivierung der Neuronen multipliziert wird. Ein Optimierer(Optimizer) schlägt bessere Gewichte vor und die Loss Funktion wird erneut berechnet. Dieser Prozess wird so lange wiederholt, bis eine

³⁶ Quelle: https://www.tutorialspoint.com/keras/keras_model_compilation.htm Stand: 04.10.2021

maximale Zahl an Iterationen ("Epochen") erreicht ist oder sich der Loss nicht mehr signifikant ändert.

5.6.1.2 Anwendung

5.6.1.2.1 Spam

```
1 Epoch 1/500
2 4/4 [=====] - 2s 292ms/step - loss: 0.9774 - categorical_accuracy: 0.2773 - val_loss: 0.2526 - val_categorical_accuracy: 0.6667
3 Epoch 2/500
4 4/4 [=====] - 2s 269ms/step - loss: 0.1951 - categorical_accuracy: 0.7883 - val_loss: 0.1688 - val_categorical_accuracy: 0.8333
5 Epoch 3/500
6 4/4 [=====] - 2s 276ms/step - loss: 0.1282 - categorical_accuracy: 0.7825 - val_loss: 0.1327 - val_categorical_accuracy: 0.8750
7 Epoch 4/500
8 4/4 [=====] - 2s 255ms/step - loss: 0.0894 - categorical_accuracy: 0.8090 - val_loss: 0.1192 - val_categorical_accuracy: 0.8333
9 Epoch 5/500
10 4/4 [=====] - 2s 270ms/step - loss: 0.0688 - categorical_accuracy: 0.7871 - val_loss: 0.0731 - val_categorical_accuracy: 0.7708
11 Epoch 6/500
12 4/4 [=====] - 2s 276ms/step - loss: 0.0430 - categorical_accuracy: 0.7837 - val_loss: 0.0473 - val_categorical_accuracy: 0.7500
13 Epoch 7/500
14 4/4 [=====] - 2s 264ms/step - loss: 0.0285 - categorical_accuracy: 0.7756 - val_loss: 0.0244 - val_categorical_accuracy: 0.7708
15 Epoch 8/500
16 4/4 [=====] - 2s 267ms/step - loss: 0.0181 - categorical_accuracy: 0.7514 - val_loss: 0.0183 - val_categorical_accuracy: 0.7500
17 Epoch 9/500
18 4/4 [=====] - 2s 262ms/step - loss: 0.0150 - categorical_accuracy: 0.7434 - val_loss: 0.0112 - val_categorical_accuracy: 0.7708
19 Epoch 10/500
20 4/4 [=====] - 2s 256ms/step - loss: 0.0118 - categorical_accuracy: 0.7583 - val_loss: 0.0088 - val_categorical_accuracy: 0.7917
21 ...
```

Abbildung 27 - Colab: Output beim Training des KI-Modell 1

Man kann nachvollziehen, wie über die Iterationen die Loss Funktion minimiert wird und die Accuracy (acc, Anteil richtiger Klassifikationen) zunimmt. Nach 10 Iterationen erkennt das Modell Spammails mit einer Wahrscheinlichkeit von 79 Prozent richtig.

```
1 Epoch 490/500
2 4/4 [=====] - 2s 277ms/step - loss: 1.4209e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5969e-07 - val_categorical_accuracy: 0.9987
3 Epoch 491/500
4 4/4 [=====] - 2s 282ms/step - loss: 1.4130e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5859e-07 - val_categorical_accuracy: 0.9969
5 Epoch 492/500
6 4/4 [=====] - 2s 298ms/step - loss: 1.4053e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5749e-07 - val_categorical_accuracy: 0.9987
7 Epoch 493/500
8 4/4 [=====] - 2s 290ms/step - loss: 1.3976e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5597e-07 - val_categorical_accuracy: 0.9969
9 Epoch 494/500
10 4/4 [=====] - 2s 287ms/step - loss: 1.3899e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5487e-07 - val_categorical_accuracy: 0.9987
11 Epoch 495/500
12 4/4 [=====] - 2s 301ms/step - loss: 1.3823e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5348e-07 - val_categorical_accuracy: 0.9969
13 Epoch 496/500
14 4/4 [=====] - 2s 288ms/step - loss: 1.3747e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5251e-07 - val_categorical_accuracy: 0.9987
15 Epoch 497/500
16 4/4 [=====] - 2s 296ms/step - loss: 1.3672e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5146e-07 - val_categorical_accuracy: 0.9969
17 Epoch 498/500
18 4/4 [=====] - 2s 282ms/step - loss: 1.3597e-06 - categorical_accuracy: 0.8470 - val_loss: 2.5015e-07 - val_categorical_accuracy: 0.9987
19 Epoch 499/500
20 4/4 [=====] - 2s 289ms/step - loss: 1.3524e-06 - categorical_accuracy: 0.8470 - val_loss: 2.4929e-07 - val_categorical_accuracy: 0.9969
21 Epoch 500/500
22 4/4 [=====] - 2s 282ms/step - loss: 1.3452e-06 - categorical_accuracy: 0.8470 - val_loss: 2.4801e-07 - val_categorical_accuracy: 0.9987
```

Abbildung 28 - Colab: Output beim Training des KI-Modell Spam 2

Nachdem Training von 500 Iterationen erkennt das Modell Spam E-Mails mit einer Wahrscheinlichkeit von nahezu 99,87%.

5.6.1.2.2 Tags

```

1 Epoch 90/2000
2 4/4 [=====] - 2s 270ms/step - loss: 3.9067e-04 - categorical_accuracy: 0.8527 - val_loss: 2.6370e-05 - val_categorical_accuracy: 0.9167
3 Epoch 91/2000
4 4/4 [=====] - 2s 280ms/step - loss: 3.7625e-04 - categorical_accuracy: 0.8481 - val_loss: 2.5403e-05 - val_categorical_accuracy: 0.9167
5 Epoch 92/2000
6 4/4 [=====] - 2s 274ms/step - loss: 3.6274e-04 - categorical_accuracy: 0.8481 - val_loss: 2.4090e-05 - val_categorical_accuracy: 0.9167
7 Epoch 93/2000
8 4/4 [=====] - 2s 282ms/step - loss: 3.4915e-04 - categorical_accuracy: 0.8481 - val_loss: 2.3260e-05 - val_categorical_accuracy: 0.9375
9 Epoch 94/2000
10 4/4 [=====] - 2s 279ms/step - loss: 3.3650e-04 - categorical_accuracy: 0.8608 - val_loss: 2.2742e-05 - val_categorical_accuracy: 0.9583
11 Epoch 95/2000
12 4/4 [=====] - 2s 269ms/step - loss: 3.2454e-04 - categorical_accuracy: 0.8619 - val_loss: 2.2211e-05 - val_categorical_accuracy: 0.9583
13 Epoch 96/2000
14 4/4 [=====] - 2s 263ms/step - loss: 3.1316e-04 - categorical_accuracy: 0.8619 - val_loss: 2.1006e-05 - val_categorical_accuracy: 0.9583
15 Epoch 97/2000
16 4/4 [=====] - 2s 257ms/step - loss: 3.0184e-04 - categorical_accuracy: 0.8608 - val_loss: 1.9990e-05 - val_categorical_accuracy: 0.9167
17 Epoch 98/2000
18 4/4 [=====] - 2s 269ms/step - loss: 2.9101e-04 - categorical_accuracy: 0.8400 - val_loss: 1.9336e-05 - val_categorical_accuracy: 0.9167
19 Epoch 99/2000
20 4/4 [=====] - 2s 271ms/step - loss: 2.8090e-04 - categorical_accuracy: 0.8400 - val_loss: 1.8616e-05 - val_categorical_accuracy: 0.9167
21 Epoch 100/2000
22 4/4 [=====] - 2s 270ms/step - loss: 2.7133e-04 - categorical_accuracy: 0.8400 - val_loss: 1.7942e-05 - val_categorical_accuracy: 0.9167
23 ...

```

Abbildung 29 - Colab: Output beim Training des KI-Modell Tag 1

Auch hier kann man erkennen, mit zunehmender Anzahl an Iterationen die Loss Funktion minimiert wird und die Accuracy (Anteil richtiger Klassifikationen) zunimmt. Nach Ca. 100 Iterationen erkennt das Modell die Tags mit nahezu 91%, das bedeutet das an dieser Stelle deutlich mehr Trainings Durchläufe stattfinden müssen, um eine Genauigkeit nahe 100% zu erreichen.

```

1 Epoch 1990/2000
2 4/4 [=====] - 2s 310ms/step - loss: 1.9521e-08 - categorical_accuracy: 0.8539 - val_loss: 7.0431e-09 - val_categorical_accuracy: 0.9583
3 Epoch 1991/2000
4 4/4 [=====] - 2s 328ms/step - loss: 1.9486e-08 - categorical_accuracy: 0.8539 - val_loss: 7.0233e-09 - val_categorical_accuracy: 0.9375
5 Epoch 1992/2000
6 4/4 [=====] - 2s 307ms/step - loss: 1.9450e-08 - categorical_accuracy: 0.8481 - val_loss: 7.0274e-09 - val_categorical_accuracy: 0.9583
7 Epoch 1993/2000
8 4/4 [=====] - 2s 332ms/step - loss: 1.9416e-08 - categorical_accuracy: 0.8539 - val_loss: 7.0101e-09 - val_categorical_accuracy: 0.9167
9 Epoch 1994/2000
10 4/4 [=====] - 2s 314ms/step - loss: 1.9380e-08 - categorical_accuracy: 0.8412 - val_loss: 7.0116e-09 - val_categorical_accuracy: 0.9375
11 Epoch 1995/2000
12 4/4 [=====] - 2s 312ms/step - loss: 1.9345e-08 - categorical_accuracy: 0.8446 - val_loss: 6.9939e-09 - val_categorical_accuracy: 0.9167
13 Epoch 1996/2000
14 4/4 [=====] - 2s 307ms/step - loss: 1.9311e-08 - categorical_accuracy: 0.8435 - val_loss: 6.9948e-09 - val_categorical_accuracy: 0.9583
15 Epoch 1997/2000
16 4/4 [=====] - 2s 331ms/step - loss: 1.9274e-08 - categorical_accuracy: 0.8516 - val_loss: 6.9615e-09 - val_categorical_accuracy: 0.9375
17 Epoch 1998/2000
18 4/4 [=====] - 2s 325ms/step - loss: 1.9239e-08 - categorical_accuracy: 0.8493 - val_loss: 6.9803e-09 - val_categorical_accuracy: 0.9583
19 Epoch 1999/2000
20 4/4 [=====] - 2s 339ms/step - loss: 1.9204e-08 - categorical_accuracy: 0.8539 - val_loss: 6.9334e-09 - val_categorical_accuracy: 0.9375
21 Epoch 2000/2000
22 4/4 [=====] - 2s 341ms/step - loss: 1.9163e-08 - categorical_accuracy: 0.8470 - val_loss: 6.9292e-09 - val_categorical_accuracy: 0.9583

```

Abbildung 30 - Colab: Output beim Training des KI-Modell Tag 2

Nach Abschluss der 2000 Iterationen, ist auch den Graphen zu entnehmen, dass sich der Genauigkeitswert bei ca 96 Prozent einpendelt.

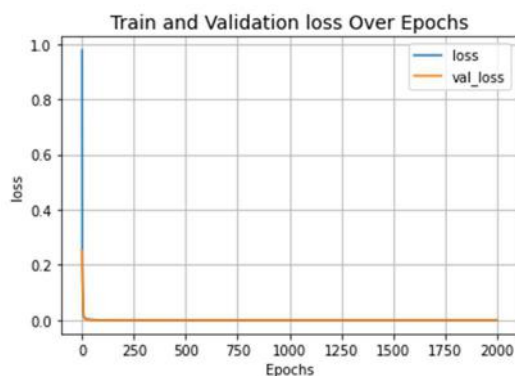


Abbildung 32 Lernkurve: KI-Modell

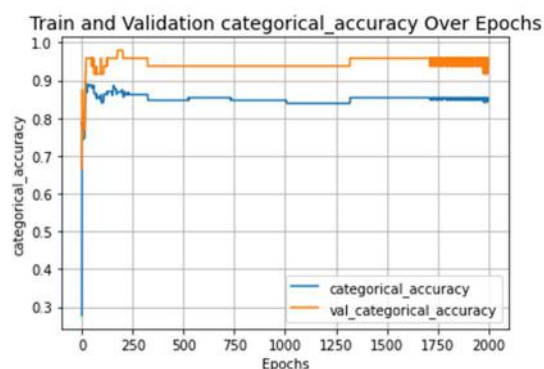


Abbildung 31 Lernrate beim Training des KI-Modells

Ebenfalls kann man den Graphen mit den Verläufen der Lernkurven zu erkennen, dass die Lernrate zu Beginn sehr steil ist und danach stetig abflacht, das bedeutet, dass auch mit zunehmender Anzahl an Iterationen, keine merklich bessere Genauigkeit erreicht werden kann. Dazu wären mehr Trainingsdaten notwendig.

5.6.1.2.3 Feedback-Schleife: Ablauf

Der Hauptmechanismus zum Trainieren unseres Modells ist die Feedback-Schleife. Die grundlegende Idee hierbei ist, dass man dem Modell die Texte zeigt und die Vorhersagen zu den einzelnen Texten notieren. Anschließend wird die Korrektheit der Vorhersagen überprüft. Im nächsten Schritt werden die Algorithmen des Modells angepasst, mit dem Ziel, eine größere Anzahl korrekter Aussagen zu erhalten. Danach wiederholt sich dieser Vorgang. Dieser Vorgang wird als Feedback-Schleife bezeichnet. Je größer also die Anzahl der Epochen eines neuronalen Netzes ist, desto länger wurde es trainiert. In der Praxis wird nicht jeder Algorithmus des Netzes von Hand angepasst, diese Feinjustierung der verschiedenen Algorithmen innerhalb eines neuronalen Netzes werden automatisiert in der Feedback-Schleife durchgeführt. Hierfür gibt es eine Vielzahl von Algorithmen, welche diese Aufgabe für uns übernehmen und zum Trainieren neuronaler Netze eingesetzt werden.

6 Test-Phase

Wie bereits in der Durchführungsphase beim Iterativen Training beschrieben, wird nach jedem Trainingsdurchlauf eine Auswertung der Ergebnisse durchgeführt, und daraufhin die Parameter des KI-Modells weiter angepasst. Somit werden in jeder Entwicklungsphase bereits Tests durchgeführt. In der Test-Phase werden keine Parameter mehr verändert. Stattdessen wird hier auf Grundlage der vorhergegangenen Anpassungen, aus der Trainingsphase untersucht, ob das Neuronale Netz etwas gelernt hat. Dazu werden den Input-Neuronen Reize übergeben und geprüft, welchen Output das Netz zurückgibt. Es wird zwischen zwei verschiedenen Arten von Reizen bzw. Inputs unterschieden: Erstens, Ausgangsreize, also bekannte Input Daten bei denen überprüft wird, ob das Netz die Trainingsdaten erfasst hat. Zweitens, Neue Reize durch das Testen von Neuen Inputs kann man feststellen, ob das Netz in der Lage ist, Aufgaben zu lösen. Oder anders gesagt, lässt sich das im Training erlernt, generalisieren und auf neue Reize Übertragen?

6.1 Vergleich Ist und Soll der Klassifizierung

6.1.1 Überschneidung: "Iteratives trainieren"

Sollte bei der Ausgabe des Outputs dann immer noch fehlerhafte Werte enthalten sein, muss eine Phase zurück gegangen werden, in die Trainingsphase. Dort können dann wieder die Parameter des Netzes angepasst werden, eine erneute Iteration wird durchgeführt (Feedback-Schleife). Dieser Kreislauf wird so lange durchgeführt, bis die Vorhersagen des Modells den Vorgaben des Projektes entsprechen.

6.1.2 Abschluss Betrachtung des Trainings

Nachdem die Vorhersagen der KI-Modelle für Spam und Tag bei jeweils 98 und 96 Prozent liegen, kann das Training im Test-System abgeschlossen werden. Jetzt wird der das Modelle ins Produktiv-System überführt. Dort wird das Trainierte Modell dann mit neu eintreffenden E-Mails getestet. Auch dort entsprechen die Vorhersagen der KI-Modelle den Vorgaben. Die Vorhersagen für Spam und Tag ermöglichen es Semi die E-Mails zu verschieben. Spam wird im Produktiv-System zuverlässig erkannt. Tags werden auch richtig vorhergesagt bei ca. 80% aller E-Mails. Bei 20% der E-Mails kann kein Tag erkannt werden, an dieser Stelle wird die E-Mail von Semi als nicht klassifizierbar markiert.

6.1.3 Auswertung

Da Spam zuverlässig erkannt wird, kann dieser Teil des Projektes als erfolgreich abgeschlossen angesehen werden. Die Erkennung von Themen bzw. Tags findet im Produktiv-System mit 80% statt, da den Graphen zur Lernkurve des Modells zu entnehmen war, dass auch mit zunehmender Anzahl an Iterationen auch keine höhere Genauigkeit erreicht werden kann, müssen mehr Trainingsdaten gesammelt und das Modell damit erneut trainiert werden.

6.1.4 Soll/Ist Vergleich

Abschließend betrachtet wurden alle im Pflichtenheft definierten Anforderungen umgesetzt. Das Anfangs erstellte Projektplan konnte ebenfalls erfüllt werden. In der Tabelle Soll/Ist-Vergleich ist der benötigte Zeitaufwand gegenübergestellt worden. Es ist dieser Tabelle zu entnehmen, dass es leichte Abweichungen von dem geplanten zeitlichen Ablauf gab. Da viele Arbeitsschritte in der Entwurfsphase des Projektes zu einer guten Planung für die Projektdurchführung geführt haben, lief die Codierung in der Durchführungsphase fast reibungslos ab. Einzige bedeutende Abweichung ist die geringe Bedeutung der Schlüsselwort Extraktion für die Vorhersage der Modelle und die umfangreiche Textvorverarbeitung. Sodass die bei der Durchführungsphase

eingesparte Zeit in der Testphase genutzt und zum Erstellen der Dokumentation genutzt werden konnte. Somit konnte das Projekt planmäßig innerhalb von 70h umgesetzt werden.

Projektphase	Soll	Ist	Differenz
Analyse	6h	6h	0h
Projektplanung	6h	6h	0h
Design/Entwurfs-Phase	4h	4h	0h
Projektdurchführung	28h	24h	-4h
Test-Phase	9h	11h	+2h
Projektabschluss	17h	19h	+2h
Gesamt:	70h	70h	70h

Tabelle - Zeitlich Abweichungen bei der Projektdurchführung

6.2 Test-Szenarien mit Trainingsdaten (Black- & White-Box-Test)

6.2.1 Test-System (white-box-test)

Im Test-System fand das Iterative Training der KI-Modelle mit den Trainingsdaten statt. Damit ist das Test-System als „Whiteboxtest“ zu betrachten. In dieses System hatte der Auftragnehmer vollen Einblick. Durch die agile Entwicklung wurden viele Einzelschritte durchgeführt, kontrolliert und getestet. Von dem Verbindungstest der Datenbank, über die einzelnen Schritte bei der Textvorverarbeitung zu dem iterativen Training der KI-Modelle.

6.2.2 Produktiv-System (black-box-test)

Zusätzlich dazu wurde das Modell im Produktiv-System mit Neuen Input Daten also den unbekannten E-Mails getestet. Dieser Schritt kann als „Blackboxtest“ betrachtet werden, weil die Daten unbekannt sind. Auch hier entsprachen die Vorhersagen der Modelle den Vorgaben.

7 Projektabschluss

7.1 Übergabe des Projektes an den Kunden

7.1.1 Übergabe

Nachdem die Anwendung fertig entwickelt und implementiert wurde, wurde sie vom Ausbilder abgenommen. Da sowohl die Funktion als auch der Umgang mit den KI-Modellen durch die Entwicklerdokumentation bekannt war, lief die Endabnahme problemlos. Aufgrund der Cloud-Architektur des Systems und der Tatsache das KI-Modell im Test-System ausreichend trainiert und getestet wurden, mussten nur die Schnittstellen ausgetauscht werden, um einen reibungslosen Ablauf im Produktiv-System zu gewährleisten. Vor der finalen Implementierung des KI-Modells im Produktivsystem wurde abschließend ein Code-Review mit den Sylvenstein Entwicklern durchgeführt.

7.2 Fazit

7.2.1 Ausblick

Dieser Abschnitt behandelt die weiteren Möglichkeiten, die aufgrund der Durchführung dieses Projekts möglich sind.

7.2.1.1 Ausblick

Nachdem nun im Produktiv-System nun für einen bestimmten Anteil keine Themen gefunden werden konnten, müssten neu Trainingsdaten hinzugefügt werden. Dazu sollte von Sylvenstein neue Daten in der Mitarbeiter Tabelle ergänzt werden, und passende E-Mails in der E-Mail-Tabelle hinzugefügt werden. Und es müsste eine weitere Iteration des Trainings stattfinden. Daraufhin könnte das Modell wieder ins Produktiv-System überführt werden, damit ist eine Skalierbarkeit und Zukunftsfähigkeit des Projektes für unbekannte Texte gewährleistet. Das bedeutet, dass dieses Projekt automatisiert arbeiten kann, bis ein unbekannter Fall eintritt ab diesem Zeitpunkt ist ein Menschlicher Eingriff in die weitere Entwicklung erforderlich, damit ist eine Skalierbarkeit und Zukunftsfähigkeit des Projektes für unbekannte Texte gewährleistet. Alle zukünftigen Erweiterungsmöglichkeiten sind Bestandteil der Entwicklerdokumentation.

7.2.2 Wissenserwerb (Know how)

Ebenfalls ist dieses Projekt als Blaupause für die Anwendung von überwachtem maschinellem Lernen (Supervised Machine Learning) zu betrachten. Eine Abwandlung des Modells ist einfach möglich, dazu muss nur das Modell kopiert und mit anderen Daten trainiert werden. Ein weiterer Anwendungsbereich für diese Technik wäre zum Beispiel, die Analyse von Kommentaren und Bewertungen auf Kunden-Webseiten (bspw. E-Commerce). Ebenfalls stellt dieses Projekt einen Einstieg für Sylvenstein in dem Bereich der Data-Science da. Da in diesem Projekt Python verwendet wurde gab es viele Bibliotheken, die sich für vielfältig einsetzbar im Bereich der Datenanalyse eignen. Beispielsweise wurden Pandas und SQLAlchemy verwendet, um große Datenmengen aus der MySQL-Datenbank im KI-Modell verarbeiten zu können. Außerdem wurden verschiedene Bibliotheken eingesetzt, um Daten mithilfe von Diagrammen visualisieren zu. Diese Visualisierung ist nicht notwendig für ein KI-Modell, aber sehr hilfreich bei der Entwicklung, dem Training und der Auswertung des Modells. Auf diese Weise soll der Umgang und die Visualisierung von großen Datenmengen mithilfe von neuronalen Netzen durch dieses Projekt einen Wissenserwerb für Sylvenstein darstellen.

7.2.3 Speichern und Laden von KI-Modellen

Die Modelle in diesem Projekt wurden mithilfe von TensorFlow entwickelt. Mithilfe der Objektserialisierung ist es möglich das trainierte Modell zur Laufzeit zu speichern und an anderer Stelle zu laden. Nachdem das Modell einmal kompiliert wurde, ist eine Speicherung möglich:

```
1 # Save the entire model as a SavedModel.
2 !mkdir -p saved_model
3 model.save('path/my_model')
```

Abbildung 33 - Quellcode: KI-Modell Speichern

Das Laden des Modells ist ebenfalls möglich wie der folgende Quellcode zeigt:

```
1 IN:
2 Load Modelmodel = tf.keras.models.load_model('path/my_model')
3
4 OUT:
5 Model: "sequential_13"
6
7 Layer (type)                Output Shape                Param #
8 -----
9 dense_21 (Dense)            (None, 512)                 2203136
10
11 dense_22 (Dense)            (None, 256)                 131328
12
13 dense_23 (Dense)            (None, 31)                  7967
14
15 Total params: 2,342,431
16 Trainable params: 2,342,431
17 Non-trainable params: 0
18
```

Abbildung 34 - Quellcode: KI-Modell Laden

7.2.4 Deploy Modell in Colab via FastAPI

7.2.5 Webhosting mit FastAPI

Mit FastAPI ist es ermöglicht es in Colab ein KI-Modell zu entwickeln und zu trainieren und von dort aus direkt zu implementieren. FastAPI, ermöglicht es die Colab Laufzeitumgebung als API zu verwenden. Damit kann dem ML-Modell ein Text direkt übergeben werden, in Colab analysiert werden, eine Vorhersage getroffen werden und die Antwort zurückgegeben werden. Dieses Vorgehen würde die Implementierung von KI-Modelle in zukünftigen Projekten vereinfachen.

7.3 Erstellen der Dokumentation

Die Dokumentation der KI-Modell setzt sich aus einem Benutzerhandbuch und der Projektdokumentation zusammen, die beide vom Autor erstellt werden. Die Entwicklerdokumentation³⁷ besteht aus einer Anleitung zur Implementierung verschiedener Bestandteile der Textvorverarbeitung, dem Design und der Anpassung der KI-Modelle und Dokumentation der Schnittstellen. Damit Mitarbeiter, unabhängig vom Auftragnehmer weiter mit

³⁷ Ein Ausschnitt befindet sich im Anhang

dem erworbenen Wissen arbeiten können. In der Projektdokumentation werden die einzelnen Projektphasen und ihre Vorgänge entsprechend erläutert.

8 Anhang

8.1 Literaturverzeichnis

Anon., 2021. *Wikipedia*. [Online]

Available at: https://en.wikipedia.org/wiki/Sentiment_analysis

Anon., kein Datum *Wikipedia*. [Online]

Available at: <https://de.wikipedia.org/wiki/Flexion>

[Zugriff am 04 10 2021].

8.2 Tabellenverzeichnis

Tabelle 1 – Mitarbeitertabelle	8
Tabelle 2 – Beispielhafte Abbildung eines Arbeitspaketbeschreibung.....	13
Tabelle 3 - Ressourcenplanung	15
Tabelle 4 - Monatliche Kosten: Betriebsmittel	17
Tabelle 5 - Berechnung Entwicklungskosten.....	17
Tabelle 6 Textblob: Deutsche anhand eines Textes Lernen.....	25
Tabelle 7 Erläuterung: Bag of Words	26
Tabelle 8 - Vorhersage/Tags.....	37
Tabelle 9 - Auflistung Softwareresourcen.....	VII

8.3 Abbildungsverzeichnis

Abbildung 1 – Trainingsdaten aufteilen	9
Abbildung 2 - Gantt Diagramm	15
Abbildung 3 - Iteratives Trainieren.....	16
Abbildung 4 Grafische Darstellung der Amortisierung	18
Abbildung 5 - UML Aktivitätsdiagramm: Grundkonzept.....	20
Abbildung 6-Colab: install SQLAlchemy	21
Abbildung 7 -Colab: Connection String für SQLAlchemy	21
Abbildung 8 - Funktion: Import von Daten mit SQLAlchemy.....	22
Abbildung 9 - Funktion: Export von Daten mit SQLAlchemy	22
Abbildung 10 - MySQL: Aufbau der Mitarbeitertabelle	23
Abbildung 11 - Skizze: Bedeutung der Label der Trainingsdaten	27

Abbildung 12 - Neuronales Netz: Vereinfachte Darstellung	28
Abbildung 13 Quellcode: Textanreicherung	33
Abbildung 14 Quellcode: Colab Installation Textblob.....	33
Abbildung 15 - Quellcode: Stammformreduktion.....	34
Abbildung 16 - Quellcode: Lemmatisierung	34
Abbildung 17 - Quellcode: Schlüsselwort Extraktion	35
Abbildung 18 - Quellcode: Schlüsselsätze extrahieren	36
Abbildung 19 - Quellcode: BoG Zuordnung	37
Abbildung 20 Quellcode: Modell Spam.....	37
Abbildung 21 - Quellcode: Modell Tags	37
Abbildung 22 Quellcode: KI-Modell kompillieren	39
Abbildung 23 - Colab: Output beim Training des KI-Modell 1	40
Abbildung 24 - Colab: Output beim Training des KI-Modell Spam 2	40
Abbildung 25 - Colab: Output beim Training des KI-Modell Tag 1.....	41
Abbildung 26 - Colab: Output beim Training des KI-Modell Tag 2.....	41
Abbildung 27 Lernrate beim Training des KI-Modells.....	41
Abbildung 28 Lernkurve: KI-Modell	41
Abbildung 29 - Quellcode: KI-Modell Speichern	46
Abbildung 30 - Quellcode: KI-Modell Laden	46
Abbildung 31 – Grafische Darstellung einer Sigmoid-Funktion	III
Abbildung 32 - Mathematische Darstellung der Sigmoid-Funktion.....	III
Abbildung 33- Verbindungstest MySQL	VII
Abbildung 34 - Erstellungs-Skript: Mitarbeiter Tabelle.....	VIII
Abbildung 35 - Erstellungs-Skript: E-Mail Tabelle	VIII

8.4 Glossar

8.4.1 SCRUM

Scrum ist ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung. Es wurde ursprünglich in der Softwaretechnik entwickelt, ist aber davon unabhängig. Scrum wird inzwischen in vielen anderen Bereichen eingesetzt. Es ist eine Umsetzung von Lean Development für das Projektmanagement.

8.4.2 Sigmoid-Funktion

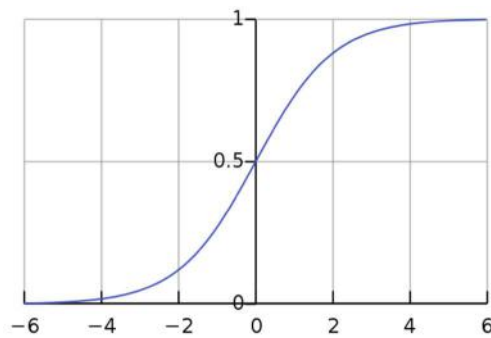


Abbildung 35 – Grafische Darstellung einer Sigmoid-Funktion

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} = 1 - S(-x)$$

Abbildung 36 - Mathematische Darstellung der Sigmoid-Funktion

8.4.3 CSV

Das Dateiformat CSV steht für (englisch) Comma-separated values und beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.

8.4.4 Textblob

Textblob ist eine Python-Bibliothek zur Textvorverarbeitung. Sie bietet eine API für gängige Aufgaben des NLP: wie Part-of-Speech-Tagging, Extraktion von Substantivphrasen, Sentiment-Analyse und mehr.

8.4.5 Sentiment-Analyse

(Anon., 2021)(Quelle:“ https://en.wikipedia.org/wiki/Sentiment_analysis“)

8.4.6 Colaboratory

Mit Colaboratory oder kurz "Colab" kann Python-Code im Browser schreiben und ausführen. Colab-Notebooks haben einen eigenen Linux Kernel, in dem Python und die gängigen Bibliotheken zur Daten-Wissenschaft vorinstalliert sind. Zugriffe auf die Kommandozeile des Notizbuches sind über die Codezellen möglich, und benötigte Bibliotheken können mit "pip install" einfach installiert werden. Außerdem bietet Colab damit auch die Möglichkeiten KI-Modell in Colab zu entwickeln, trainieren, und auch zu Implementieren. KI-Modelle können mit Pickle oder Joblib gespeichert und wieder aufgerufen werden, um vorhersagen zu machen. So kann das fertig trainierte Modell beispielsweise in andere Projekte importiert werden. Andererseits kann auch die laufzeit-Umgebung von Colab verwendet werden, um das KI-Modell über eine API zur Verfügung zu stellen. Auf diese Weise kann das KI-Modell im Notebook wie eine ganz normale öffentlich Webmethode aufgerufen werden, dazu würde sich beispielsweise die fastAPI-Bibliothek eignen.

- Keine Konfiguration erforderlich
- Kostenlosen Zugriff auf GPUs

(Quelle: "<https://colab.research.google.com/>")

8.4.7 Objektserialisierung

Bei der Datenspeicherung handelt es sich bei der Serialisierung um die Übersetzung von Datenstrukturen oder Objektzuständen in ein Format, das gespeichert oder später übertragen und rekonstruiert werden kann.

(Quelle: "<https://de.acervolima.com/pickle-python-objektserialisierung/>")

8.4.8 Naive Bayes

Naive Bayes ist ein Klassifizierungsansatz, der das Prinzip der bedingten Unabhängigkeit der Klassen aus dem Bayes-Theorem übernimmt. Das bedeutet, dass das Vorhandensein eines Merkmals keinen Einfluss auf das Vorhandensein eines anderen Merkmals in Bezug auf die Wahrscheinlichkeit eines bestimmten Ergebnisses hat, und dass jeder Prädiktor den gleichen Einfluss auf das Ergebnis hat. Es gibt drei Arten von Naive Bayes-Klassifikatoren: Multinomial Naive Bayes, Bernoulli Naive Bayes und Gaussian Naive Bayes. Diese Technik wird hauptsächlich bei der Textklassifizierung, der Spam-Erkennung und bei Empfehlungssystemen eingesetzt.

8.4.9 Lineare Regression

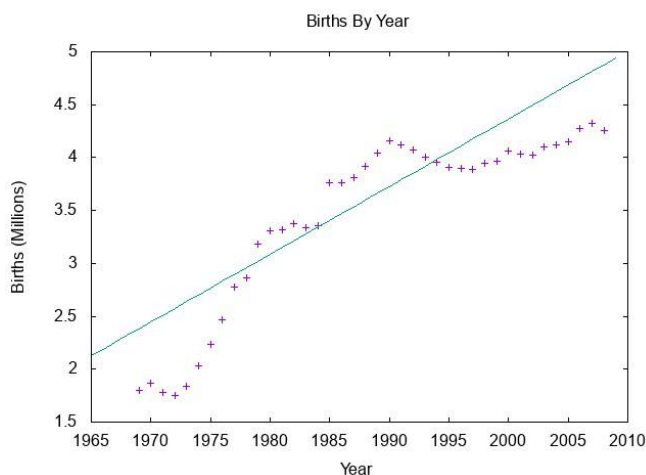


Abbildung 37- Lineare Regression

Die lineare Regression wird verwendet, um die Beziehung zwischen einer abhängigen Variablen und einer oder mehreren unabhängigen Variablen zu ermitteln und wird in der Regel dazu genutzt, Vorhersagen über zukünftige Ergebnisse zu treffen. Wenn es nur eine unabhängige Variable und eine abhängige Variable gibt, spricht man von einer einfachen linearen Regression. Wenn die Anzahl der unabhängigen Variablen zunimmt, spricht man von multipler linearer

Regression. Bei jeder Art von linearer Regression wird versucht, eine Linie der besten Anpassung zu zeichnen, die mit der Methode der kleinsten Quadrate berechnet wird. Im Gegensatz zu anderen Regressionsmodellen ist diese Linie jedoch gerade, wenn sie in ein Diagramm eingezeichnet wird.

8.4.10 Logistische Regression

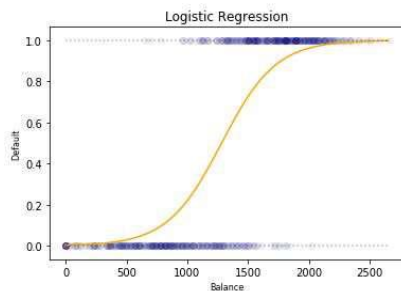


Abbildung 38- Logistische Regression

Während die lineare Regression eingesetzt wird, wenn die abhängigen Variablen kontinuierlich sind, wird die logistische Regression gewählt, wenn die abhängigen Variablen kategorisch sind, d. h. sie haben binäre Ausgänge, wie "wahr" und "falsch" oder "ja" und "nein". Während beide Regressionsmodelle versuchen, Beziehungen zwischen Dateninputs zu verstehen, wird die logistische Regression hauptsächlich zur Lösung von binären Klassifizierungsproblemen verwendet, z. B. zur Identifizierung von Spam.

8.4.11 Support-Vektor-Maschine (SVM)

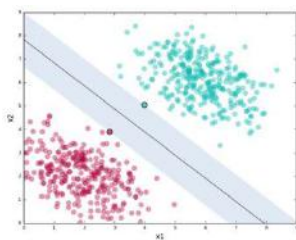


Abbildung 39 - Support Vektor Maschine

Eine Support-Vektor-Maschine ist ein beliebtes Modell des überwachten Lernens, das von Vladimir Vapnik entwickelt wurde und sowohl für die Datenklassifizierung als auch für die Regression verwendet wird. In der Regel wird es jedoch für Klassifizierungsprobleme eingesetzt, indem eine Hyperebene konstruiert wird, in der der Abstand zwischen zwei Klassen von Datenpunkten am größten ist. Diese Hyperebene wird als Entscheidungsgrenze bezeichnet und trennt die Klassen der Datenpunkte (z. B. Orangen vs. Äpfel) auf beiden Seiten der Ebene.

8.4.12 K-Nächster Nachbar(K-nearest neighbor)

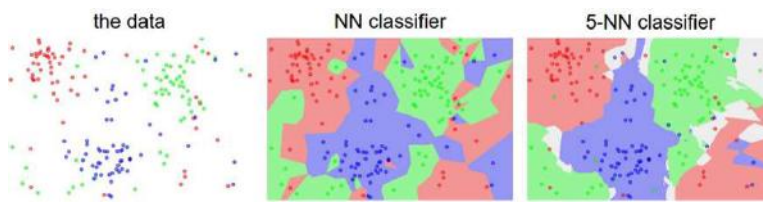


Abbildung 40 - K-Nächster

Der K-nearest neighbor, auch KNN-Algorithmus genannt, ist ein nichtparametrischer Algorithmus, der Datenpunkte auf der Grundlage ihrer Nähe und Assoziation zu anderen verfügbaren Daten klassifiziert. Dieser Algorithmus geht davon aus, dass ähnliche Datenpunkte in der Nähe zueinander gefunden werden können. Daher versucht er, den Abstand zwischen den Datenpunkten zu berechnen, in der Regel durch den euklidischen Abstand, und ordnet dann eine Kategorie auf der Grundlage der häufigsten Kategorie oder des Durchschnitts zu.

Seine Benutzerfreundlichkeit und die niedrige Berechnungszeit machen ihn zu einem bevorzugten Algorithmus für Datenwissenschaftler, aber wenn der Testdatensatz wächst, verlängert sich die Verarbeitungszeit, was ihn für Klassifizierungsaufgaben weniger attraktiv macht. KNN wird in der Regel für Empfehlungsmaschinen und Bilderkennung verwendet.

8.4.13 Zufälliger Wald (Random forest)

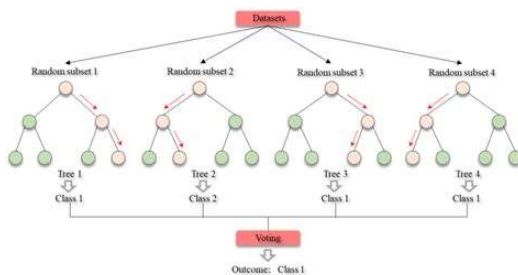


Abbildung 41 - Zufälliger Wald

Random Forest ist ein weiterer flexibler Algorithmus für überwachtes maschinelles Lernen, der sowohl für Klassifizierungs- als auch für Regressionszwecke verwendet wird. Der "Wald" bezieht sich auf eine Sammlung von unkorrelierten Entscheidungsbäumen, die dann zusammengeführt werden, um die Varianz zu verringern und genauere Datenvorhersagen zu erstellen.

8.5 Abkürzungsverzeichnis

8.6 Literaturverzeichnis

8.6.1 Tabellen & Abbildungen

8.6.1.1 Softwareressourcen

Bibliothek	Beschreibung	Quelle
SQLAlchemy	Schnittstelle	https://www.sqlalchemy.org
Pandas	Datenverarbeitung	https://pandas.pydata.org
Numpy	Datenverarbeitung	https://numpy.org
Matplotlib	Datenvisualisierung	https://matplotlib.org
pyLDAvis	Datenvisualisierung	https://pypi.org/project/pyLDAvis/
GENSIM	ML-Modell	https://radimrehurek.com/gensim/
GENSIM	summarization, Keywords	https://radimrehurek.com/gensim_3.8.3/summarization/keywords.html
NLTK	Natural Language Toolkit	https://www.nltk.org
NLTK	Extract Keywordphrases	https://gist.github.com/manmohan24nov/d7540c6f1705fba3d104e0fedb0d889b
SKLEARN	Models & Datenvisualisierung	https://scikit-learn.org/stable/index.html
SKLEARN	CountVectorizer	https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
SKLEARN	TfidfVectorizer	https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
SKLEARN	MultiLabelBinarizer	https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html
SKLEARN	Naive-Bayes	https://scikit-learn.org/stable/modules/naive_bayes.html
SKLEARN	SGDClassifier	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
SKLEARN	LogisticRegression	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
SKLEARN	LinearSVC	https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
RegEx		

Tabelle 9 - Auflistung Softwareressourcen

8.6.1.2 Test der Verschlüsselten Verbindung: MySQL

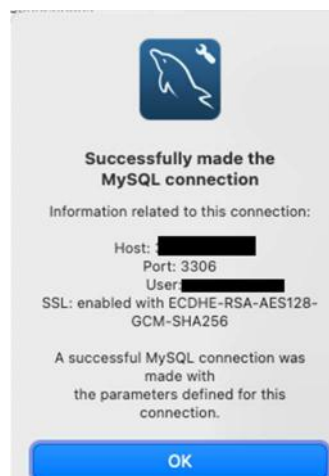


Abbildung 42- Verbindungstest MySQL

8.6.1.3 Erstellungs-Skripte für die Tabellen in der MySQL Datenbank

```
1 CREATE TABLE `Emails` (  
2   `index` bigint(20) DEFAULT NULL,  
3   `Betreff` text,  
4   `Text` text,  
5   `Von: (Name)` text,  
6   `Von: (Adresse)` text,  
7   `Von: (Typ)` text,  
8   `An: (Name)` text,  
9   `An: (Adresse)` text,  
10  `An: (Typ)` text,  
11  `CC: (Name)` text,  
12  `CC: (Adresse)` text,  
13  `CC: (Typ)` text,  
14  `BCC: (Name)` text,  
15  `BCC: (Adresse)` text,  
16  `BCC: (Typ)` text,  
17  `Abrechnungsinformationen` text,  
18  `Kategorien` text,  
19  `Reisekilometer` text,  
20  `Vertraulichkeit` text,  
21  `Wichtigkeit` text,  
22  KEY `ix_Emails_index` (`index`)  
23 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Abbildung 44 - Erstellungs-Skript: E-Mail Tabelle

```
1 CREATE TABLE `Mitarbeiter_Tags` (  
2   `id` int(11) NOT NULL,  
3   `Name` varchar(45) DEFAULT NULL,  
4   `Vorname` varchar(45) DEFAULT NULL,  
5   `Abteilung` varchar(45) DEFAULT NULL,  
6   `Thema` varchar(45) DEFAULT NULL,  
7   `Tag` varchar(45) DEFAULT NULL,  
8   PRIMARY KEY (`id`)  
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Abbildung 43 - Erstellungs-Skript: Mitarbeiter
Tabelle

Dokumenten Anhang

Entwickler Dokumentation

Es folgt ein Auszug aus der Entwicklerdokumentation:

Entwickler Dokumentation

Philipp Braun

Entwickler Dokumentation

Klassifizierung mithilfe von Künstlicher Intelligenz

Angefertigt: Philipp Braun

Notizbücher:

- | | |
|------------------------------------|---|
| 1. Schnittstelle SQLAlchemy | https://colab.research.google.com/AI_1/XXXXX |
| 2. Textvorverarbeitung | https://colab.research.google.com/AI_2/XXXXX |
| 3. KI-Modell: Spam Erkennung | https://colab.research.google.com/AI_3/XXXXX |
| 4. KI-Modell: Themen/Tag Erkennung | https://colab.research.google.com/AI_4/XXXXX |

1. Colaboratory

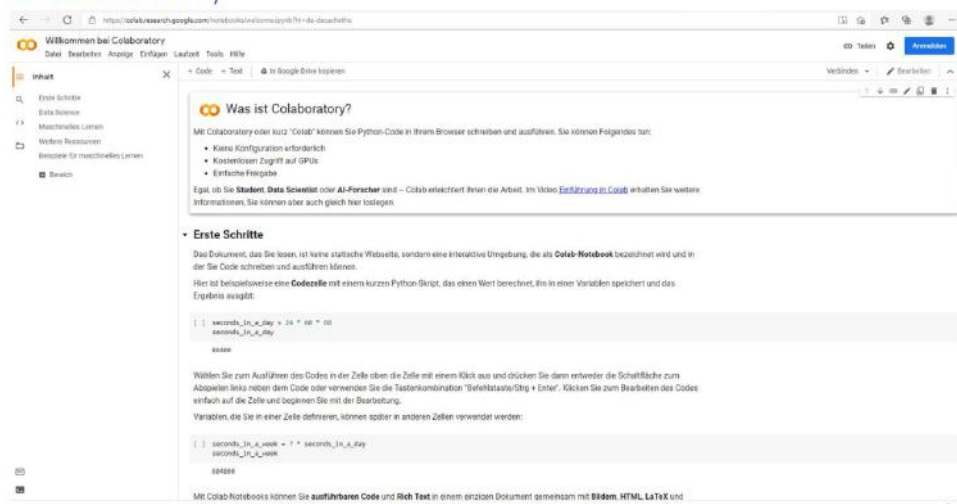


Abbildung 1 Aufbau(Colabootory kurz Colab)

Melden sie sich bei <https://colab.research.google.com/notebooks/welcome.ipynb> mit ihrem Googlekonto an und schon steht ihnen die Laufzeitumgebung von Colab zur Verfügung

2. Was ist Colaboratory?

Um genau zu sein, ist Colab eine kostenlose Jupyter-Notebook-Umgebung, die vollständig in der Cloud ausgeführt wird. Vor allem ist kein Setup erforderlich, und die von Ihnen erstellten Notizbücher können gleichzeitig von Ihren Teammitgliedern bearbeitet werden - genau so, wie Sie Dokumente in Google Text & Tabellen bearbeiten.

3. Funktion

Colab unterstützt viele gängige Bibliotheken für maschinelles Lernen, die problemlos in Ihr Notebook geladen werden können.

Entwickler Dokumentation

Philipp Braun

4. Schnittstelle SQLAlchemy

Installation

Der folgende Quellcode beschreibt die Installation von SQLAlchemy in Colab mit PIP(Python).

Abbildung 2 Quellcode: Installation SQLAlchemy in Colab

Zur Verbindung mit der Datenbank muss für SQLAlchemy ein Connection String mit folgendem Aufbau definiert werden:

```
1 engine = create_engine("dialect+driver://username:password@host:port/database")
```

Abbildung 3 -Colab: Connection String für SQLAlchemy

Zum Import und Export der Daten werden zwei Funktionen benötigt, der folgende Quellcode beschreibt die Definition beider Funktionen.

```
1 read_sql_table(tabellenName, ConnectionString)
2 engine = create_engine(ConnectionString)
3 dataframe = pandas.read_sql_table(tabellenName, engine)
4 return dataframe
```

Abbildung 4 - Funktion: Import von Daten mit SQLAlchemy

```
1 To_sql(TabellenName, ConnectionString)
2 engine = create_engine(ConnectionString)
3 dataframe.to_sql(TabellenName, con=engine)
```

Abbildung 5 - Funktion: Export von Daten mit SQLAlchemy

5. Textvorverarbeitung

Die Textvorverarbeitung in diesem Projekt besteht aus vielen kleinen Einzelschritten, die je nach Bedarf und Anwendungsfall zu einer Funktion zusammengefasst werden können. Und so auf eine Reihe, Spalte oder Zelle angewendet werden können.

Textanreicherung:

Bei der Textanreicherung werden die ursprünglichen Textdaten mit Informationen ergänzt, die Sie zuvor nicht hatten. Das bedeutet, dass in der E-Mail-Tabelle eine neue Spalte erzeugt werden soll. In dieser Spalte werden Text der E-Mail, Betreff, Absender und Empfänger zur Analyse zusammengefasst werden.

6.

```
1 # Combining Columns for single input of analysis
2 dataframe["Analysis"] = dataframe["Betreff"]
3   + dataframe["Text"]
4   + dataframe["Absender"]
5   + dataframe["Empfänger"]
6   + dataframe["Tags"]
```

7. Abbildung 6 Quellcode: Textanreicherung

Geräuschentfernung(Noise removal)

Bei der Geräuschentfernung geht es darum, Zeichen, Ziffern und Textteile zu entfernen, die bei der Textanalyse stören könnten. Geräuschentfernung ist stark vom Anwendungsfall abhängig. In Tweets können beispielsweise Sonderzeichen als Rauschen(Noise) vorkommen. Es gibt verschiedene Möglichkeiten, Geräusche zu entfernen. Dazu gehören z.B. die Entfernung von Satzzeichen, Sonderzeichen, Zahlen, HTML-Formatierung, anwendungsfallabhängige Schlüsselwörter (z.B. "WG" für Weitergeleitet), Quellcodeentfernung usw..

```
1 # def function for removing noise
2 def removeNoise(text):
3     re.sub(r'[\.\?!\,\;\:\\"RE\WG]', '', text)
4
5 # Apply function
6 dataframe['Analysis'].apply(removeNoise)
```

Abbildung 7 Quellcode: Geräuschentfernung

Rechtschreibung Korrigieren mit Textblob

Installation von Textblob:

```
1 pip install -U textblob
```

Abbildung 8 Quellcode: Colab Installation Textblob

Zum Korrigieren der Rechtschreibung soll in diesem Projekt die Python Bibliothek Textblob¹ eingesetzt werden. Textblob verwendet Statistiken über die Wortgebrauch in Sprache, um intelligente Vorschläge zu machen, welche Wörter korrigiert werden sollten. Um mit Textblob auch die deutsche Sprache Korrigieren zu können, muss eine statische Textdatei erzeugt werden. Die Datei sollte ausreichend Daten über die deutsche Sprache enthalten um Textblob ausreichend trainieren zu können². Der Folgende Code-Ausschnitt beschreibt wie diese Datei erzeugt werden kann:

```
1 from textblob.en import Spelling
2 import re
3
4 textToLower = ""
5
6 with open("germanSample.txt","r") as f1:      # Open our source file
7     text = f1.read()                          # Read the file
8     textToLower = text.lower()                # Lower all the capital letters
9
10 words = re.findall("[a-z]+", textToLower)    # Find all the words and place them into a list
11 oneString = " ".join(words)                  # Join them into one string
12
13 pathToFile = "train.txt"                     # The path we want to store our stats file at
14 spelling = Spelling(path = pathToFile)       # Connect the path to the Spelling object
15 spelling.train(oneString, pathToFile)        # Train
```

Tabelle 1 Textblob: Deutsche anhand eines Textes Lernen

Quellcode zum korrigieren der Rechtschreibung in Deutsch und Englischer Sprach:

```
def clean_blob_correct(text):
    if detect(text) == 'en':
        blob = TextBlob(text)
    elif detect(text) == 'de':
        blob = TextBlobDE(text)
    return blob.correct().split()
```

Abbildung 9 Rechtschreibung korrigieren

Stammformreduktion(Stemmin)

Stammformreduktion ist ein NLP-Prozess, der die Flexion(Veränderung)³ in Wörtern auf ihre Stammformen reduziert. Flexion ist die Veränderung eines Wortes, um verschiedene grammatikalische Kategorien wie Zeitform, Kasus, Person, Zahl, Geschlecht und Stimmung auszudrücken. Ein Wort kann also in verschiedenen Formen existieren, aber diese Formen von Wörtern im selben Text führen zu Redundanz für NLP-Operationen. Daher wird die Stammformreduktion durchgeführt, wobei solche Stämme möglicherweise nicht einmal ein gültiges Wort sind. Beispiel dafür ist der Stamm von Reduktion, reduzieren und reduziertes: "redu", was an sich kein richtiges Wort ist.

¹ Siehe Anhang: **Fehler! Verweisquelle konnte nicht gefunden werden.** Textblob

² Bei der Durchführung dieses Projektes wurde eine Textdatei mit langem deutschem Beispielttext erstellt(60.000Wörter)

³ Quelle: Wikipedia <https://de.wikipedia.org/wiki/Flexion> Stand: 04.10.2021

```
1 # Imports
2 from nltk.stem import SnowballStemmer
3
4 # def stemmer
5 snowball = SnowballStemmer(language='english')
6
7 # def stemming function
8 def snowStemmer(text):
9     stemmed_text = ' '.join([snowball.stem(word) for word in text.split()])
10    return stemmed_text.split()
```

Abbildung 10 Stammformreduktion

Lemmatisierung

Lemmatisierung ist der Prozess, bei dem der Kontext verwendet wird, um ein Wort in seine sinnvolle Basis- oder Stammform umzuwandeln. In der Linguistik ist ein Lemma ein bedeutungsvolles Basiswort oder ein Wortstamm, der die Grundlage für andere Wörter bildet. Zum Beispiel ist das Lemma der Wörter "spielen" und "gespielt" "spielen". Der Unterschied zwischen Stemming und Lemmatisierung besteht darin, dass die Stammformreduktion zu ungültigen Wörtern führt, aber lemmatisierte Wörter führen immer zu sinnvollen Wörtern. Der Grund dafür ist, dass diese Wörter mit einem Wörterbuch abgeglichen werden.

```
1 # Imports
2 from nltk.stem import WordNetLemmatizer
3
4 # def lemmatizer
5 lemmatizer = WordNetLemmatizer()
6
7 # def stemming function
8 def lemmatize(text):
9     lemmatized_text = ' '.join([lemmatizer.lemmatize(word) for word in text.split()])
10    return lemmatized_text.split()
```

Abbildung 11 Lemmatisierung

Entfernung von Stoppwörtern

Stoppwörter sind eine Gruppe von Wörtern, die in einer Sprache häufig verwendet werden. Beispiele für Stoppwörter im Englischen sind "a", "the", "is", "are" usw.. Die Intention hinter der Entfernung von Stoppwörtern ist, die wichtigen Wörter analysiert werden können, indem wir Wörter mit geringer Information aus dem Text entfernen. Wenn im Text beispielsweise "Wann ist das Banner fertig?" steht, sollte „ist“ und „das“ entfernt werden. Dies kann dadurch erreicht werden, dass alle Wörter aus einer Stoppwörterliste entfernt werden. Da deutsche und englische Texte verarbeitet werden sollen, müssen Stoppwörterlisten für Deutsch & Englisch verwendet werden.

Entwickler Dokumentation

Philipp Braun

```
# use German stop word set
stop = set(stopwords.words('german'))

# add extra example stopwords which are not in list
stop.update({'der','die','das','dass','koennen','sind','the'})

# add english stopwords to stopword-list
stop.update(set(stopwords.words('english')))

# define funktion for stop-word-removal, which can be applied to df['col']
def clean_stop(text):
    stop_free = ' '.join([word for word in text.lower().split() if word not in stop])
    return stop_free.split()
```

Abbildung 12 Stoppwortlisten und Stoppwortentfernung

Quellcode auf Dataframe anwenden:

Abbildung 13 Textvorverarbeitung auf Dataframe anwenden

8. KI-Modelle:

Beim Entwurf der Neuronalen Netze ist drauf zu achten, dass der Gesamte E-Mail-Text in beiden Modellen als Input für die Input-Layer dienen soll. Die Hidden-Layers werden in beiden Modellen erstmal gleich definiert, mit 2 Layer aus jeweils 256 Neuronen. In der Output-Layer liegt der Zentrale unterschied beider Modelle. Bei der Spam-Erkennung ist es wichtig zu beachten, dass nur ein einzelner Wert zurückgegeben wird, nämlich 0 oder 1 für Spam oder nicht Spam-Mail. Bei der Vorhersage der Tags werden zwei Output-Neuronen benötigt, jedes Neuron kann einen oder keinen Wert für ein Tag zurückgeben. Damit kann dann ein, zwei oder kein Tag zurückgegeben werden. Die Anpassungen Hidden-Layer, der Algorithmen und Aktivierungsschwellenwerte innerhalb der Neuronalen Netzes sind Teil des iterativen Trainings der Modelle. Nach jeder Iteration sollen die Parameter angepasst werden, bis die Vorhersagen der Modelle die gewünschte Genauigkeit erreicht haben.

Spam Erkennung

```
1 # Modell container
2 model = tf.keras.models.Sequential()
3
4 # Input-Layer
5 model.add(tf.keras.layers.Dense(256, input_shape=x_train.shape, activation='sigmoid'))
6 # Hidden-Layer
7 model.add(tf.keras.layers.Dense(256, activation='sigmoid'))
8 # Output-Layer
9 model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

9. Abbildung 14 Quellcode: Modell Spam

Entwickler Dokumentation

Philipp Braun

Themen/Tag Erkennung

```
1 # Modell container
2 model = tf.keras.models.Sequential()
3
4 # Input-Layer
5 model.add(tf.keras.layers.Dense(256, input_shape=x_train.shape, activation='sigmoid'))
6 # Hidden-Layer
7 model.add(tf.keras.layers.Dense(256, activation='sigmoid'))
8 # Output-Layer
9 model.add(tf.keras.layers.Dense(1, output_mode="multi_hot", activation="sigmoid"))
```

10. Abbildung 15 - Quellcode: Modell Tags

Mit `compile()` werden der Optimierer (Optimizer), die Loss Funktion und zusätzliche Metriken konfiguriert⁴. Mit `fit()` werden dem Modell die Input- und Zieldaten für das Training übergeben und mit „epochs“ die Anzahl der Iterationen festgelegt.

```
# Modell konfigurieren
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

# Modell trainieren
model.fit(x = input, y = target, epochs = 500)
```

11. Abbildung 16 Quellcode: KI-Modell kompilieren

Feedback-Schleife

Der Hauptmechanismus zum Trainieren unseres Modells ist die Feedback-Schleife. Die grundlegende Idee hierbei ist, dass man dem Modell die Texte zeigt und die Vorhersagen zu den einzelnen Texten notieren. Anschließend wird die Korrektheit der Vorhersagen überprüft. Im nächsten Schritt werden die Algorithmen des Modells angepasst, mit dem Ziel, eine größere Anzahl korrekter Aussagen zu erhalten. Danach wiederholt sich dieser Vorgang. Dieser Vorgang wird als Feedback-Schleife bezeichnet. Je größer also die Anzahl der Epochen eines neuronalen Netzes ist, desto länger wurde es trainiert. In der Praxis wird nicht jeder Algorithmus des Netzes von Hand angepasst, diese Feinjustierung der verschiedenen Algorithmen innerhalb eines neuronalen Netzes werden automatisiert in der Feedback-Schleife durchgeführt. Hierfür gibt es eine Vielzahl von Algorithmen, welche diese Aufgabe für uns übernehmen und zum Trainieren neuronaler Netze eingesetzt werden.

⁴ Quelle: https://www.tutorialspoint.com/keras/keras_model_compilation.htm Stand: 04.10.2021

12. 6. Ausblick

Speichern und Laden von KI-Modellen (Objektserialisierung)

Die Modelle in diesem Projekt wurden mithilfe von TensorFlow entwickelt. Mithilfe der Objektserialisierung ist es möglich die Trainierten Modell zur Laufzeit zu Speichern und an anderer Stelle zu laden. Nachdem das Modell einmal kompiliert wurde, ist eine Speicherung möglich:

```
1 # Save the entire model as a SavedModel.
2 !mkdir -p saved_model
3 modell.save('path/my_model')
```

Abbildung 17 - Quellcode: KI-Modell Speichern

Das Laden des Modell ist ebenfalls möglich wie der Folgende Quellcode zeigt:

```
1 IN:
2 Load Modellmodel = tf.keras.models.load_model('path/my_model')
3
4 OUT:
5 Model: "sequential_13"
6
7 Layer (type)          Output Shape          Param #
8 -----
9 dense_21 (Dense)      (None, 512)           2203136
10 -----
11 dense_22 (Dense)      (None, 256)           131328
12 -----
13 dense_23 (Dense)      (None, 31)            7967
14 -----
15 Total params: 2,342,431
16 Trainable params: 2,342,431
17 Non-trainable params: 0
18 -----
```

Abbildung 18 - Quellcode: KI-Modell Laden

Entwickler Dokumentation

Philipp Braun

Deploy Modell in Colab via FastAPI

Webhosting mit FastAPI

Mit FastAPI ist es ermöglicht es in Colab ein KI-Modell zu entwickeln und zu trainieren und von dort aus direkt zu Implementieren. FastAPI, ermöglicht es die Colab Laufzeitumgebung als API zu verwenden. Damit kann dem ML-Modell ein Text direkt übergeben werden, in Colab analysiert werden, eine Vorhersage getroffen werden und die Antwort zurückgegeben werden. Dieses Vorgehen würde die Implementierung von KI-Modelle in zukünftigen Projekten vereinfachen.

```
1 !pip install fastapi
```

Abbildung 19- Quellcode: Installation FastApi

FastAPI ist ein neues Python-basiertes Web-Framework zur Erstellung von Web-APIs. FastAPI ist schnell bei der Bereitstellung Ihrer Anwendung.

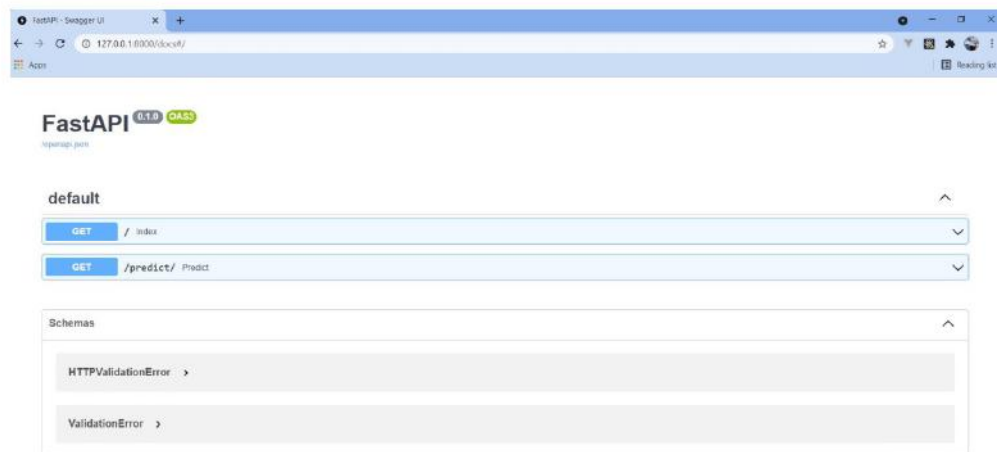


Abbildung 20 FastAPI via Lokalhost

Abnahmeprotokoll

Klassifizierung von E-Mails mithilfe von Künstlicher Intelligenz

Datum: 08.10.2021

Teilnehmer: Philipp Braun

Name	Bezeichnung	Funktion
Christian Giebel (Sylvenstein GmbH)	Auftraggeber	Projektleiter
Philipp Braun	Auftragnehmer	Teilnehmer

1 Abnahmegegenstand

Projektarbeit: Klassifizierung von E-Mails mithilfe von Künstlicher Intelligenz

ID	Bezeichnung	Version, Details
L1	KI-Modell: Spam Erkennung	1.0, Colab
L2	KI-Modell: Tags Erkennung	1.0, Colab

2 Ergebnisse der Abnahmeprüfung

Die Anwendung muss folgende Abnahmekriterien erfüllen:

1. KI: Spam erkennen
2. KI: Themen erkennen und passende Tags setzten
3. SEMI: Verknüpfung vom KI-System, Semi und Mail-Server

3 Zu erledigende Punkte

Der Teilnehmer hat noch folgende Aufgaben im Rahmen des Vertrags zu erfüllen:

ID	Task	Abgenommen
1	MySQL Datenbank als Zentrale Schnittstelle	X
1.1	Mitarbeiter Zuständigkeits-Tabelle	X
1.2	Test-System	X
1.3	Produktiv-System	X
2	Erstellen der KI-Modelle	X
2.1	Laufzeitumgebung: Colab	X
2.2	Schnittstelle zur Datenbank: SQLAlchemy	X
2.3	Neuronales Netzwerk entwickeln	X

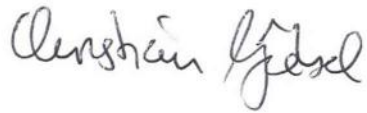

Sylvenstein

2.4	KI: Spam Erkennung	X
2.5	KI: Tag Erkennung	X
2.6	Iteratives Trainieren der Modelle	X
2.7	Export der Vorhersagen für Spam & Thema in die Datenbank	X
3	Implementierung der KI-Modelle	X
3.1	Semi: Import von Emails in Produktiv-System	X
3.2	KI: Vorhersagen treffen im Produktiv-System	X
3.3	Semi: Verschieben der E-Mails	X
3.4.1	Spam Ordner	X
3.4.2	Thema: Tag zum zuständigen Mitarbeiter	X

4 Schlussbeurteilung

Die unter Pkt. 3. genannten Punkte erfüllen die Anforderungen vollständig.

5 Genehmigung/Freigabe

Titel	Name	Unterschrift
Für den AUFTRAGGEBER	Christian Giebel	
Für den AUFTRAGNEHMER	Philipp Braun	

Persönliche Erklärung

Erklärung des Prüfungsteilnehmers / der Prüfungsteilnehmerin:

Ich versichere durch meine Unterschrift, dass ich das betriebliche Projekt und die dazugehörige Dokumentation selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Dortmund, 20.10.2021



Ort und Datum

Unterschrift

Erklärung des Ausbildungsbetriebes / Praktikumsbetriebes:

Wir versichern, dass der betriebliche Auftrag wie in der Dokumentation dargestellt, in unserem Unternehmen realisiert worden ist.

0231 9525354

Christian Giebel

Telefon/Durchwahl

Ansprechpartner

Dortmund, 20.10.2021



Ort und Datum

Unterschrift und **Firmenstempel**

Die unterschriebene „Persönliche Erklärung“ ist der Online-Version nur hinzuzufügen, wenn keine ausgedruckten Exemplare der Dokumentation angefordert worden sind!