HAW HAMBURG

BACHELOR THESIS
Lasse Frederik Schulz

Föderales Lernen am Beispiel der Kategorisierung von Mails

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Lasse Frederik Schulz

Föderales Lernen am Beispiel der Kategorisierung von Mails

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Peer Stelldinger
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 04. März 2025

Lasse Frederik Schulz

Thema der Arbeit

Föderales Lernen am Beispiel der Kategorisierung von Mails

Stichworte

Maschinelles Lernen, Föderales Lernen, Datenschutz, Klassifizierung, Differential Privacy, Secure Aggregation

Kurzzusammenfassung

Ziel dieser Arbeit ist es, die praktische Umsetzung von föderalem Lernen zu untersuchen und die Vor- und Nachteile im Vergleich zum zentralisierten maschinellen Lernen zu betrachten. Dazu wird föderales Lernen mit dem Flower Framework anhand der Kategorisierung von Mails als Anwendungsfall umgesetzt. Zur Umsetzung der Kategorisierung werden Naive Bayes-Klassifikatoren, LSTM-Modelle und der Einsatz von Random Forest betrachtet. Zusätzlich zum föderalen Lernen werden Differential Privacy und Secure Aggregation zum Schutz der Trainingsdaten eingesetzt. Das Training beim föderalen Lernen wird mit dem zentralen Lernen in verschiedenen Szenarien verglichen. Als Resultat der Arbeit wird erfolgreich aufgezeigt, wie die Umsetzung eines Anwendungsfalls beim föderalen Lernen erfolgen kann. Zudem kommt heraus, dass föderales Lernen allgemein Modelle mit ähnlicher Effektivität zu zentral trainierten Modellen trainieren kann, auch unter Einsatz von Differential Privacy und Secure Aggregation, und dass das Training durch die Verteilung des Trainings schneller abläuft.

Lasse Frederik Schulz

Title of Thesis

Comparison of federated learning and centralized machine learning based on the categorization of emails

Keywords

Machine learning, federated learning, data privacy, classification, differential privacy, secure aggregation

Abstract

The goal of this thesis is to evaluate the practical implementation of federated learning and to examine the advantages and disadvantages in comparison to centralized machine learning. For this purpose, federated learning is implemented with the Flower Framework using the categorization of mails as a use case. Naive Bayes classifiers, LSTM models and the use of Random Forest are considered for the implementation of the categorization. In addition to federated learning, differential privacy and secure aggregation are used to protect the training data. The training with federated learning is compared with centralized learning in different scenarios. The result of the work is a successful demonstration of how a use case can be implemented in federated learning. It is also shown that federated learning can generally train models with similar effectiveness to centrally trained models, even when using differential privacy and secure aggregation, and that the training is faster due to the distribution of the training.

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **ix**

1 Einleitung **1**
1.1 Motivation 1
1.2 Zielsetzung 1
1.3 Aufbau der Arbeit 3

2 Grundlagen **4**
2.1 Maschinelles Lernen 4
2.2 Föderales Lernen 6
2.2.1 Arten des föderalen Lernens 7
2.2.2 Ablauf des föderalen Lernens nach Federated Averaging 9
2.2.3 Angriffsvektoren beim föderalen Lernen 10
2.3 Datenschutz 12
2.4 Schutzmechanismen beim föderalen Lernen 14
2.4.1 Differential Privacy 14
2.4.2 Secure Aggregation 16
2.5 Techniken zur Textdarstellung 19
2.5.1 Word2Vec 19
2.6 Klassifikationsverfahren 21
2.6.1 Naive Bayes 22
2.6.2 Random Forest 23
2.6.3 Long short-term memory 24

3 Anwendungsfall **26**

4 Verwandte Arbeiten/Projekte **28**

5 Eingesetzte Techniken und Verfahren **31**

6 Implementierung **36**
6.1 Datengrundlage 36
6.1.1 Verwendete Datensätze 36
6.1.2 Vorbereitung der Daten 39
6.1.3 Word2Vec-Modell 41
6.2 Architektur 42
6.3 Umgebung der Clients und des Servers 44
6.4 Weitere Anmerkungen zur Umsetzung 46

7 Messungen und Ergebnisse **48**
7.1 Vorgehen bei den Messungen 48
7.2 Szenarien 51
7.2.1 Szenario 1: Ausgangsszenario 51
7.2.2 Szenario 2: Föderales Lernen mit Secure Aggregation 56
7.2.3 Szenario 3: Föderales Lernen mit Differential Privacy 57
7.2.4 Szenario 4: Föderales Lernen mit Differential Privacy & Secure Aggregation 61
7.2.5 Szenario 5: Zentrales Lernen mit geringer Datenmenge 62
7.2.6 Szenario 6: Verringerte Anzahl an Clients 63
7.2.7 Szenario 7: Variation der Datenmenge beim föderalen Lernen . . . 65

8 Diskussion der Ergebnisse **66**

9 Fazit **71**

10 Ausblick **73**

Literaturverzeichnis **76**

A Anhang **82**
A.1 Verwendete Hilfsmittel 82

Selbstständigkeitserklärung **83**

Abbildungsverzeichnis

2.1 Beispielhafter Ablauf beim föderalen Lernen mit zentralen Aggregator . . 9
2.2 Ablauf der Secure Aggregation [5] 17
2.3 CBOW-Architektur von Word2Vec [32] 20
2.4 Skip-gram-Architektur von Word2Vec [32] 21
2.5 Vereinfachte Darstellung eines LSTM-Modells [8] 25

6.1 Überblick der Spam-Mails der einzelnen verwendeten Mail-Datensätze . . 37
6.2 Verteilung der Mail-Längen in Enron-Spam 41
6.3 Verteilungssicht der Umgebung zur Auswertung beim föderalen Lernen . . 43
6.4 Klassendiagramm zur Implementierung des Anwendungsfalls 45
6.5 LSTM-Modell für den Anwendungsfall 47

7.1 Effektivität (F1-Score) der LSTM-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen) 52
7.2 Effizienz (Rechenzeit) der LSTM-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen) 54
7.3 Vergleich der Effektivität bei unterschiedlichen Epsilon-Werten beim LSTM-Modell (Durchschnitt von fünf Durchläufen) 59
7.4 Verlauf der Effektivität bei unterschiedlichen Epsilon-Werten beim LSTM-Modell (Durchschnitt von fünf Durchläufen) 60
7.5 Vergleich der Effektivität bei unterschiedlichen Epsilon-Werten beim Naive Bayes-Modell (Durchschnitt von fünf Durchläufen) 61
7.6 Verlauf der Effektivität von zehn Durchläufen bei Epsilon 0,3 beim Naive Bayes-Modell 62
7.7 Effektivität (F1-Score) der LSTM-Klassifikation beim zentralen Lernen mit geringer Datenmenge im Vergleich zum föderalen Lernen (Durchschnitt von fünf Durchläufen) 63

7.8 Effektivität (F1-Score) der Naive Bayes-Klassifikation beim zentralen Lernen mit geringer Datenmenge im Vergleich zum föderalen Lernen (Durchschnitt von fünf Durchläufen) 64
7.9 Effektivität (F1-Score) der LSTM-Klassifikation beim föderalen Lernen mit unterschiedlicher Datenmenge (Durchschnitt von fünf Durchläufen) . . 65

Tabellenverzeichnis

5.1 Zusammenfassung (als F1-Score) der Ergebnisse von „Comparison of Deep and Traditional Learning Methods for Email Spam Filtering“ [36] 32

6.1 Überblick der einzelnen Mail-Datensätze 39

7.1 Szenario 1: Messwerte zur Effektivität der LSTM-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen) 52
7.2 Szenario 1: Messwerte zur Effektivität der Naive Bayes-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen) 53
7.3 Szenario 1: Größe der Datensätze in MB 55
7.4 Szenario 1: Maximale Arbeitsspeichernutzung 56
7.5 Szenario 6: Effektivität des LSTM-Modells beim föderalen Lernen mit verringerter Clientanzahl (Durchschnitt von fünf Durchläufen) 65

A.1 Verwendete Hilfsmittel und Werkzeuge 82

1 Einleitung

Im folgenden Kapitel wird zunächst die Motivation, die Zielsetzung und der Aufbau der Arbeit erklärt, um als Einstieg einen Überblick über das Vorhaben dieser Arbeit zu geben. Zudem soll das Kapitel die Relevanz des Ziels, welches in der Arbeit erreicht werden soll, zeigen.

1.1 Motivation

Maschinelles Lernen als Teilbereich der Künstlichen Intelligenz ist ein immer wichtigeres Gebiet in der heutigen Informatik, welches zunehmend an Bedeutung gewinnt, da es in vielen Bereichen erfolgreich angewandt wird und immer bessere Ergebnisse liefert. Es wird verwendet, um Prozesse zu automatisieren, neue Erkenntnisse aus Daten zu gewinnen oder allgemein Projekte umzusetzen, die mit klassischer Programmierung nicht realisierbar sind. Dabei ist die Verwendung von Daten ein wichtiges Thema, da je nach Anwendungsfall eventuell auch persönliche oder andere schützenswerte Daten zum Training verwendet werden. Dies ist vor allem dann der Fall, wenn etwa maschinelles Lernen in der öffentlichen Verwaltung eingesetzt werden soll. Zudem erfordert das Training mit vielen Daten einen hohen Rechenaufwand, weshalb ein zentraler Rechner beim Training schnell an Grenzen stößt. Dabei bietet sich föderales Lernen als mögliche Lösung der Probleme an, um ein verteiltes Training von Modellen zu ermöglichen und einen besseren Schutz der Daten zu erreichen.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, anhand der Kategorisierung von Mailnachrichten als Anwendungsfall für föderales Lernen zu untersuchen, wie föderales Lernen praktisch umgesetzt

werden kann und welche Vorteile und Nachteile sich daraus im Vergleich zum zentralisierten maschinellen Lernen ergeben.

Teil der Arbeit ist es als eines der Hauptziele von föderalem Lernen auch, die individuellen Daten in der Umsetzung zu schützen, weshalb weitere Techniken zum Schutz der Daten eingesetzt werden.

Zum Erreichen des Ziels werden primär bestehende Softwarebibliotheken und Methoden verwendet und nur, wenn es notwendig ist, werden eigene Implementierungen der verwendeten Algorithmen und Techniken umgesetzt. Auch steht eine umfassende Optimierung der Modelle für den Anwendungsfall und des Trainingsprozesses nicht im Fokus der Arbeit und wird daher nicht vollständig untersucht.

Konkreter werden zur Erreichung des Ziels verschiedene Fragen betrachtet:

- Wie sieht eine geeignete Architektur zur Umsetzung von föderalem Lernen für die Kategorisierung von Mails aus?
- Wie **effektiv** ist die Klassifizierung von Mails mit föderalem Lernen im Vergleich zum zentralen Lernen gemessen an **Precision, Recall, Accuracy und F1-Score** der Klassifikation?
- Wie wirkt sich föderales Lernen auf die **Effizienz** gemessen an der benötigten **Rechenzeit** beim Training und der gesamten **Laufzeit** im Vergleich zum zentralen Lernen aus?
- Gibt es Unterschiede in der Effektivität und Effizienz zwischen verschiedenen Klassifizierungstechniken beim föderalen Lernen im Vergleich zum zentralen Lernen?
- Führt föderales Lernen zu einer Reduktion der Rechenzeit bei der Inferenz im Vergleich zum zentralen Lernen?
- Wie wirken sich weitere Schutzmechanismen zum föderalen Lernen (wie Differential Privacy und Secure Aggregation) auf die Effizienz und Effektivität aus?
- Wie viele Trainingsdaten sind notwendig, um beim föderalen Lernen eine gleiche Effektivität bei der Klassifizierung wie beim zentralen Lernen zu erreichen?
- Wie wirkt sich die Anzahl der Clients beim föderalen Lernen auf die Effizienz und Effektivität aus?
- Wie stark wird das Netzwerk beim föderalen Lernen belastet?
- Welche Vorteile beim Datenschutz bietet föderales Lernen zusammen mit den genutzten Schutzmechanismen im Vergleich zum zentralen Lernen?

1.3 Aufbau der Arbeit

Das Ziel der Arbeit ist durch die Einleitung der Arbeit festgelegt. Die Arbeit befasst sich darüber hinaus mit den Grundlagen der Konzepte, die im Kontext der Arbeit relevant sind. Hierbei wird sich bereits besonders auf die Konzepte konzentriert, welche für die spätere Umsetzung relevant sind und zum allgemeinen Verständnis der Arbeit benötigt werden. Auch umfasst dieser Abschnitt bereits eine Betrachtung des Datenschutzes im Kontext des föderalen Lernens. Darauf wird der spezifische Anwendungsfall und dessen Eignung für die Arbeit genauer erläutert und es werden verwandte Arbeiten vorgestellt. Im Kapitel „Eingesetzte Techniken und Verfahren“ wird erklärt, welche von den in den Grundlagen vorgestellten Techniken und Verfahren in der Arbeit eingesetzt werden und warum diese, auch unter Berücksichtigung des Anwendungsfalls, ausgewählt wurden. Im Anschluss werden Details zu den Daten und zur Implementierung des Anwendungsfalls beschrieben und alles drumherum, was für die Umsetzung benötigt wird. Im Kapitel darauf wird die Umsetzung in einzelnen Szenarien untersucht. Abschließend werden die Ergebnisse der Arbeit in den verschiedenen Szenarien hinsichtlich der einzelnen Fragen zur Zielsetzung der Arbeit diskutiert und danach wird allgemein zum zentralen Ziel der Arbeit ein Fazit gezogen. Darauf erfolgt noch ein kurzer Ausblick auf mögliche Weiterentwicklungen und alternative Ansätze.

2 Grundlagen

Im vorherigen Kapitel wurde der Hintergrund und die Zielsetzung der Arbeit genauer erläutert. Zur Umsetzung der Arbeit werden verschiedene Techniken, Verfahren und Konzepte angewandt. Im folgenden Kapitel werden nun die Grundlagen der Techniken, Verfahren und Konzepte erklärt, um ein besseres Verständnis für diese zu bekommen. In den folgenden Abschnitten sind sowohl grundlegende Konzepte für die Arbeit als auch solche, welche für die Umsetzung des Anwendungsfalls benötigt werden, aufgeführt und erklärt. Daneben wird auch auf den Datenschutz beim föderalen Lernen eingegangen, was auch ein Teil der Zielsetzung der Arbeit ist. Im späteren Verlauf der Arbeit wird noch erläutert, welche dieser Konzepte für die Umsetzung des Anwendungsfalls konkret relevant sind.

2.1 Maschinelles Lernen

Das föderale Lernen basiert auf dem maschinellen Lernen, weshalb zunächst geklärt wird, was überhaupt maschinelles Lernen ist. Das maschinelle Lernen ist ein Teilbereich der Künstlichen Intelligenz. Der Bereich der Künstlichen Intelligenz befasst sich nach Russel und Norvig zunächst allgemein damit, Maschinen zu entwickeln, die selbst berechnen können, wie sie effektiv und sicher in verschiedenen Bereichen handeln können [34, S.1].

Das maschinelle Lernen befasst sich nun damit, wie ein Computer (hier als Agent bezeichnet) aus einer Umgebung lernen kann, um ein bestimmtes Problem zu lösen. Konkreter beobachtet ein Agent dazu Daten (die Umgebung) und trainiert aus dieser Beobachtung ein Modell, welches Annahmen über Zusammenhänge der Daten und dem zu lösenden Problem trifft. Dieses Modell kann dann genutzt werden, um das Problem zu lösen. Sinn dahinter ist, dass ein solches Modell bei einer großen oder gar unendlichen Anzahl an Zuständen der Umgebung trotzdem ein Problem lösen kann. Das Modell stellt somit eine Funktion dar, die für jede Eingabe auch eine passende Ausgabe liefert. Auch

kann maschinelles Lernen dazu genutzt werden, Probleme zu lösen, für die ein Programmierer möglicherweise keinen konkreten Algorithmus definieren und umsetzen kann [34, S.651].

Dabei wird beim maschinellen Lernen zwischen verschiedenen Arten, wie ein solches Modell gelernt wird, um ein Problem zu lösen, unterschieden [3, S.3]:

- **Überwachtes Lernen (Supervised Learning)**

  Beim überwachten Lernen sind Eingabe und Ausgabe zum Zeitpunkt des Trainings bereits bekannt. Die Trainingsdaten, mit denen ein Modell trainiert werden soll, bestehen aus Beispielergebnissen der Ausgabe und den dazugehörigen Eingabevektoren. Zu den Hauptproblemklassen beim überwachten Lernen gehört etwa die Klassifizierung und die Regression. Bei der Klassifizierung wird ein Eingabevektor einer Klasse aus einer endlichen Menge an Klassen durch das trainierte Modell zugeordnet. Es gibt somit einen endlichen Bereich an möglichen Ausgabevektoren, denen ein Eingabevektor zugeordnet werden kann. Bei der Regression hingegen wird zu einem Eingabevektor eine oder mehrere kontinuierliche Variablen in der Ausgabe zugeordnet. Die Anzahl der möglichen Ausgabevektoren ist hierbei hingegen eventuell unendlich groß oder zumindest sehr groß.

- **Unüberwachtes Lernen (Unsupervised Learning)**

  Beim unüberwachten Lernen ist im Vergleich zum überwachten Lernen zum Zeitpunkt des Trainings nur die Eingabe und keine Ausgabe bekannt. Die Trainingsdaten bestehen somit nur aus Eingabevektoren ohne Beispielergebnisse der Ausgabe. Auch beim unüberwachten Lernen kann zwischen verschiedenen Problemklassen unterschieden werden. Einer der bekanntesten ist das Clustering. Beim Clustering wird aus den Beispieldaten, welche die Eingabevektoren umfassen, nach einer zuvor definierten Methode bestimmt, wie ähnlich ein Eingabevektor zu einem anderen ist. Durch dieses Ähnlichkeitsmaß werden anschließend eigenständig Gruppen (Cluster) aus Eingabevektoren gebildet. Eine weitere Problemklasse ist die Dichteschätzung. Bei der Dichteschätzung wird die Verteilung der Daten im Raum der Eingabe geschätzt. Zur Visualisierung kann unüberwachtes Lernen schließlich noch zur Dimensionsreduktion genutzt werden, um die Anzahl der Dimensionen des Eingabevektors auf weniger und dadurch möglicherweise besser visualisierbare zu reduzieren.

- **Bestärkendes Lernen (Reinforcement Learning)**

Schließlich gibt es noch das bestärkende Lernen beim maschinellen Lernen. Beim bestärkenden Lernen erhält ein Modell typischerweise eine Umgebung als Eingabevektor und gibt eine optimale Aktion für die Umgebung als Ausgabevektor zurück. Das Modell lernt durch das Ausprobieren, welche Aktionen in welcher Situation gut oder schlecht sind, da es für jede Aktion eine Belohnung oder Bestrafung erhält und abhängig von der erhaltenen Belohnung/Bestrafung automatisch angepasst wird. Die Trainingsdaten bestehen somit aus der Umgebung, der gewählten Aktion und der erhaltenen Belohnung. Genauer betrachtet wird beim bestärkenden Lernen bei der Wahl der Aktion zwischen Exploration und Exploitation unterschieden. Zu Beginn des Trainings wird beim bestärkenden Lernen allgemein im Sinne der Exploration zufällig eine Aktion gewählt und die Belohnung beobachtet. Im späteren Verlauf des Trainings übernimmt dann immer mehr die Exploitation und es wird stattdessen die beste Aktion gewählt, die bisher in der jeweiligen Situation die höchste Belohnung erzielt hat.

Daneben gibt es weitere spezifischere Arten beim maschinellen Lernen, die Kombinationen aus den vorherigen drei sind, wie das Semi-Supervised Learning, oder auf diesen Grundarten aufbauen, wie das Multi-Task Learning oder das Transfer Learning, welche hier jedoch nicht weiter ausgeführt werden [15, S.243, S.244, S.536].

Das klassisch zentrale maschinelle Lernen, auch in seinen unterschiedlichen Arten, hat jedoch auch einige Nachteile, die zu Problemen in der Anwendung führen können. Je nach Art des maschinellen Lernens und des Vorgehens kommt es zum Overfitting auf die Trainingsdaten, vor allem, wenn die Trainingsdaten nicht besonders divers sind und keine breite Auswahl möglicher Eingaben abdecken. Außerdem kann maschinelles Lernen zum Training eines Modells sehr rechenaufwändig werden und hohe Datenmengen benötigen. Die Daten müssen dazu zudem erst an einem zentralen Ort gebracht werden, was weiteren Aufwand erfordert und auch zu Problemen beim Datenschutz führen kann.

2.2 Föderales Lernen

Föderales Lernen (kurz FL) bietet Lösungen für einen Teil der Probleme, die beim zentralen maschinellen Lernen auftreten können. Das föderale Lernen befasst sich damit, wie beim maschinellen Lernen Modelle verteilt trainiert werden können, ohne die Trainingsdaten dazu an einem zentralen Ort zu sammeln. In einem klassischen Szenario beim

maschinellen Lernen können große Mengen an Daten zum Training vorhanden sein. Klassisch zentrales Lernen stößt dabei aber an Grenzen, da die Geschwindigkeit der Verarbeitung durch die Systemressourcen der Rechner begrenzt ist, um diese Daten zum Training zu verarbeiten. Föderales Lernen bietet sich dazu als mögliche Lösung an, um verteiltes Lernen von Modellen zu ermöglichen. McMahan et al. beschreiben föderales Lernen als Verfahren, bei dem mehrere Clients, die alle lokale Daten zum Training besitzen, ein gemeinsames Modell trainieren, indem lokal berechnete Aktualisierungen aggregiert werden [24]. Clients können beliebige Geräte sein, die auch in den Systemressourcen variieren, wie etwa Smartphones oder IoT-Geräte. Die Daten bleiben dabei bei den Clients und werden nicht untereinander oder mit einer zentralen Stelle geteilt. Ein zentraler Server dient dabei oft als Aggregator und Koordinator beim föderalen Lernen. Dieser fragt einzelne Clients in jeder Runde nach Aktualisierungen, wobei nicht alle Clients immer teilnehmen müssen. Die Clients trainieren dabei lokale Modelle und senden aber nur die Gewichte/Parameter an den zentralen Server. Die Berechnung der Modelle wird somit zu den Daten gebracht, statt die Daten zum zentralen Server, um ein Modell zu berechnen.

Aus dem Vorgehen beim föderalen Lernen ergeben sich nach McMahan et al. einige Vorteile gegenüber dem zentralen Lernen:

- Die Kommunikationslast durch das Training sinkt deutlich, da nicht mehr sämtliche Trainingsdaten übertragen werden müssen. McMahan et al. beobachtet dabei eine Reduzierung der Anzahl der benötigten Kommunikationsrunden im Vergleich zum stochastischen Gradientenabstieg um den Faktor 10 bis 100.
- Die Daten müssen nicht zentral gesammelt werden, was vorteilhaft für den Datenschutz ist.
- Die Systemressourcennutzung beim Training kann besser auf viele Clients verteilt werden, wodurch das Training insgesamt schneller ablaufen kann.

2.2.1 Arten des föderalen Lernens

Nun gibt es beim föderalen Lernen nicht nur ein besonderes Szenario, auf das sich spezialisiert wird. Die Originalarbeit von McMahan et al. konzentriert sich vor allem auf einen Anwendungsbereich mit mobilen Clients. Kairouz et al. unterscheiden weiter in „Advances and open problems in federated learning“ zwischen zwei zentralen Arten des

föderalen Lernens [19, S.5 f.]. Diese beiden Arten werden zur weiteren Unterscheidung in dieser Arbeit einmal genauer erläutert.

Cross-device federated learning

Das Szenario von McMahan et al. beschreiben Kairouz et al. als „Cross-device federated learning“. Beim Cross-device federated learning hat jeder Client meist eine moderate Datenmenge und ist ein mobiles Gerät wie ein Smartphone oder ein IoT-Gerät. Daher sind die Clients in dem Fall in der Leistung eher beschränkt. Die Clients sind auch nicht unbedingt zuverlässig und antworten somit nicht immer auf Anfragen beim Trainingsprozess oder sind gar nicht erst verfügbar, da z.B. ein Smartphone nicht immer erreichbar ist. Auch können die Clients in der Kommunikation eingeschränkt sein, da etwa die Internetverbindung langsam ist. Diese Einschränkungen sind jedoch in dem Szenario kein Problem, da es eine sehr große Anzahl an Clients gibt und in jeder Trainingsrunde nur ein kleiner Teil am Training teilnimmt.

Cross-silo federated learning

Als „Cross-silo federated learning“ hingegen wird das föderale Lernen mit deutlich weniger Clients bezeichnet. Dabei ist ein Client oft ein Unternehmen, ein Standort oder eine Abteilung. Die wenigen Clients haben hingegen deutlich mehr Daten als beim Cross-device federated learning. Sie sind zudem zuverlässiger und ihnen stehen eine mehr Systemressourcen zur Verfügung. Auch die Kommunikation zwischen Server und Clients ist meist zuverlässiger und schneller. Durch das insgesamt stabilere System und der geringeren Anzahl an Clients nehmen beim Cross-silo federated learning meist alle Clients an einer Trainingsrunde teil.

Vertikales und horizontales föderales Lernen

Allgemein wird beim föderalen Lernen auch zwischen vertikalen und horizontalen föderalen Lernen unterschieden. Die Einteilung in vertikal/horizontal bezieht sich dabei auf die Verteilung der Daten über die Clients [19, S.15]. Beim horizontalen föderalen Lernen haben alle Clients die gleichen Features, aber unterschiedliche Trainingsdaten. Beim vertikalen föderalen Lernen hingegen haben die Daten einzelner Clients jeweils unterschiedliche Features.

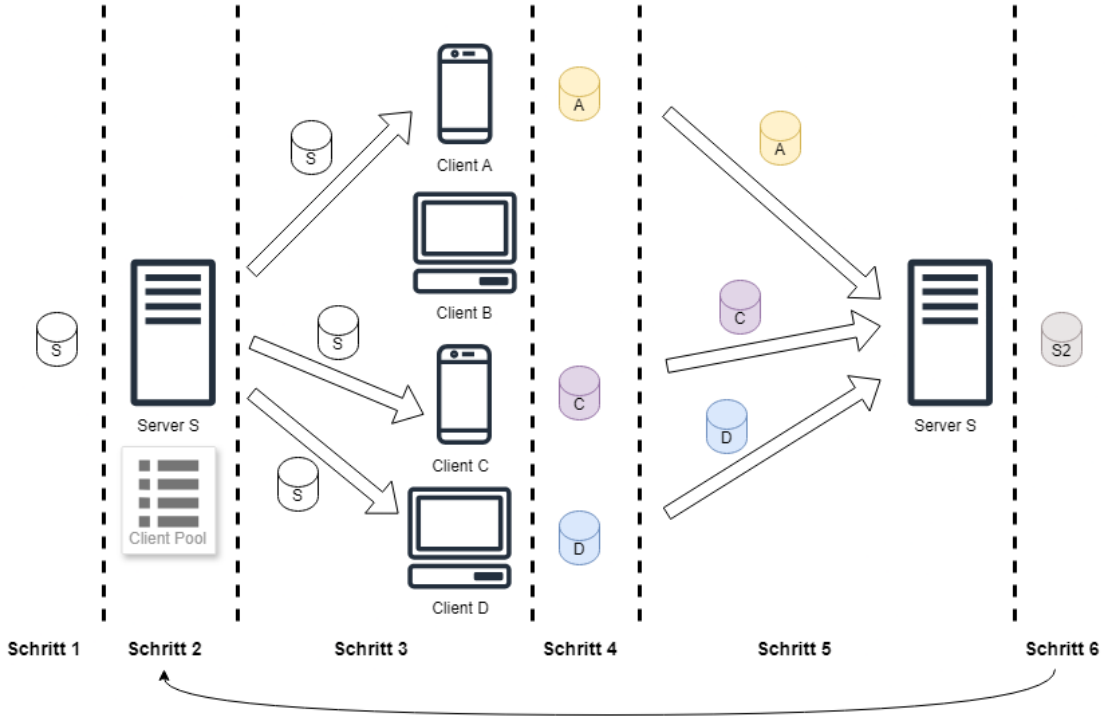

Abbildung 2.1: Beispielhafter Ablauf beim föderalen Lernen mit zentralen Aggregator

2.2.2 Ablauf des föderalen Lernens nach Federated Averaging

Kairouz et al. beschreiben auch, wie ein allgemeiner Ablauf des Trainings eines Modells beim föderalen Lernen aussehen kann, basierend auf der von McMahan et al. vorgestellten Methode des Federated Averaging (FedAvg) [19, S.8] (Abbildung 2.1 stellt diesen Ablauf dar):

1. Ein zentraler Server initialisiert ein Modell, welches trainiert werden soll.
2. Der zentrale Server wählt einzelne Clients aus dem Pool verfügbarer Clients aus, die als Teil der aktuellen Trainingsrunde das Modell trainieren sollen (die ausgewählten Clients müssen dazu für das Training verfügbar sein).
3. Die ausgewählten Clients erhalten vom zentralen Server die aktuellen Modellparameter und übernehmen diese als lokales Modell.
4. Die Clients trainieren das Modell lokal mit ihren lokalen Daten und senden die aktualisierten Modellparameter an den zentralen Server zurück.

5. Der zentrale Server aggregiert die Modellparameter der Clients und erhält über eine definierte Methode eine gemeinsame Aktualisierung für das globale Modell.
6. Der zentrale Server aktualisiert das globale Modell mit der aggregierten Aktualisierung und startet ggf. eine neue Trainingsrunde (Schritt 2).

Im Falle von FedAvg führen die Clients einen Schritt des Stochastischen Gradientenabstiegs (SGD) mit ihren lokalen Daten durch und der zentrale Server aggregiert die aktualisierten Modellparameter der Clients gewichtet nach der Anzahl der Daten [24, S.4]. Neben Federated Averaging gibt es für den Ablauf beim föderalen Lernen verschiedene weitere Varianten, die sich primär bei der Aggregation unterscheiden, hier jedoch nicht mehr weiter erläutert werden.

2.2.3 Angriffsvektoren beim föderalen Lernen

Sowohl beim zentralen als auch beim föderalen Lernen gibt es verschiedene Angriffsvektoren, für die Modelle beim oder nach dem Training anfällig sind. Im Folgenden wird ein Überblick über die Angriffsvektoren beim föderalen Lernen gegeben, da es auch Teil des föderalen Lernens und Teil dieser Arbeit ist, vor Angriffen auf die Daten beim Training und damit die Modelle zu schützen.

Shirvani et al. beschreiben dazu in „Federated Learning: Attacks, Defenses, Opportunities, and Challenges“ mögliche Angriffsvektoren beim föderalen Lernen [37, S.5-7]. Sie unterscheiden allgemein zwischen Angriffen auf die Sicherheit eines Modells und Angriffen auf den Schutz der Daten. Angriffen auf die Sicherheit eines Modells versuchen, die Qualität oder Ergebnisse des Modells zu beeinflussen. Angriffen auf den Schutz der Daten versuchen hingegen, Trainingsdaten aus den Modellen zu extrahieren.

Angriffe auf die Sicherheit eines Modells sind im Kontext der Arbeit eher weniger relevant, da die einzelnen Clients, die zum Training beitragen, als vertrauenswürdig gelten. Trotzdem werden Angriffe auf die Sicherheit kurz betrachtet: Shirvani et al. beschreiben als zwei grundlegende Angriffe auf die Sicherheit Sybil-Angriffe und Byzantinische Angriffe. Bei Sybil-Angriffen kontrolliert ein Angreifer mehrere Clients und beeinflusst das Training, indem diese alle ähnliche Ergebnisse liefern und damit das globale Modell in eine Richtung lenken. Bei Byzantinischen Angriffen hingegen schicken Clients zufällige Aktualisierungen für das Modell an den zentralen Server, um die Konvergenz des

Trainings zu verhindern. Auch wird allgemein unterschieden, wie das Ergebnis des Trainings beeinflusst wird. Angreifer könnten dabei an den Trainingsdaten ansetzen und diese verändern oder direkt das Modell manipulieren.

Angriffe auf den Schutz der Daten hingegen spielen im Kontext der Arbeit eine zentrale Rolle. Zu den relevantesten Arten von Angriffen auf den Schutz der Daten zählen:

1. Membership Inference Attack: Hierbei versucht ein Angreifer herauszufinden, ob ein bestimmter Eintrag in den Trainingsdaten enthalten ist. Ein Angreifer kennt dabei nur eventuell das Modell oder kann nur das Modell befragen und wertet anhand der Ergebnisse aus, ob ein Eintrag in den Trainingsdaten enthalten ist. Dies wird dem Angreifer ermöglicht, da beim maschinellen Lernen oftmals Overfitting auftritt und ein Modell so etwa bestimmte Trainingsdaten besonders sicher klassifizieren kann.

2. Reconstruction Attack: Dieser Angriff versucht die Trainingsdaten zu rekonstruieren. Ein Angreifer kennt dabei die Modellparameter und versucht damit von den Ausgaben eines Modells auf die Trainingsdaten zurückzuschließen (im Kontext des Anwendungsfalls der Arbeit könnte ein Angreifer z.B. Teile einzelner E-Mails und einzelne Informationen aus E-Mails bis hin zu ganzen E-Mails rekonstruieren).

3. Attribute Inference Attack: Bei einem solchen Angriff versucht ein Angreifer, Eigenschaften der Trainingsdaten insgesamt aus dem Modell abzuleiten. Diese Eigenschaften sind nicht direkt ein Teil der Trainingsdaten und müssen sich auch nicht auf einzelne Einträge in den Daten beziehen sondern eher auf die gesamten Trainingsdaten (im Kontext des Anwendungsfalls der Arbeit könnte das z.B. das Verhältnis von Spam/Ham Mails in den genutzten Trainingsdaten sein).

4. Linkage Attack: Ein Angreifer rekonstruiert durch einen anderen Angriff Attribute oder ganze Einträge der Trainingsdaten. Der Angreifer verknüpft die Informationen, die dabei gewonnen wurden, mit Informationen aus einem anderen Datensatz, den der Angreifer bereits hat, um z.B. anonymisierte Einträge Personen zuzuordnen [13, S.8,10].

5. Quantity Estimation Attack: Hierbei versucht ein Angreifer, die Anzahl der Trainingsdaten mit einem bestimmten Label durch das Modell zu bestimmen.

6. Feature Inference Attack: Bei diesem Angriff versucht ein Angreifer beim vertikalen föderalen Lernen, die Features eines Clients direkt aus den trainierten Modell auszulesen. Ein Angreifer hat dabei oft Zugriff auf die Aktualisierungen des Modells,

welche von den Clients zur Aggregation gesendet werden und versucht daraus auf die Features zu schließen.

7. Free-Rider Angriffe: Bei dieser Art von Angriff, beteiligt sich ein Angreifer als Client am Training, trägt aber selber nichts zum Training bei. Stattdessen erhält der Angreifer die Modellparameter für den eigenen Nutzen, ohne etwas beizutragen.

2.3 Datenschutz

Im Kontext des maschinellen Lernens spielt der Datenschutz eine wichtige Rolle, da zum Training von Modellen oft große Mengen an Daten benötigt werden, die eventuell schützenswert sind, da sie z.B. personenbezogene Daten enthalten. Hierzu gibt es auch verschiedene gesetzliche Regelungen, die den Umgang mit Daten vorgeben und auch beim maschinellen Lernen eingehalten werden müssen. EU-weit gilt hierzu die Datenschutz-Grundverordnung (DSGVO) und in Deutschland das Bundesdatenschutzgesetz (BDSG). Die DSGVO hat dabei Vorrang vor dem BDSG und das BDSG vor weiteren Datenschutzgesetzen. Die DSGVO gilt für alle Bereiche (öffentlich und nichtöffentlich), wohingegen das BDSG nur für nichtöffentliche Stellen gilt und für öffentliche Stellen des Bundes angewandt wird. Für öffentliche Stellen der Bundesländer gelten die jeweiligen Landesdatenschutzgesetze. Daneben haben Kirche und öffentlicher Rundfunk noch eigene Datenschutzgesetze [7, S.14 f.].

Die DSGVO gibt zum Datenschutz fünf Grundprinzipien vor, die bei der Verarbeitung von Daten eingehalten werden müssen [7, S.21 f.]:

1. Rechtmäßigkeit: Die Verarbeitung personenbezogener Daten (Artikel 4 der DSGVO definiert personenbezogene Daten als Informationen, die direkt oder indirekt zur Identifikation einer natürlichen Person genutzt werden können [7, S.141 f.]) ist nur dann erlaubt, wenn diese auf einer Rechtsgrundlage basiert, daher durch eine gesetzliche Regelung oder Einwilligung der betroffenen Personen erlaubt wird.

2. Transparenz: Es muss umfassend über die Verarbeitung von Daten informiert werden, daher z.B. in welchem Umfang und zu welchem Zweck Daten verarbeitet werden.

3. Datenminimierung: Es darf nur das Minimum an benötigten Daten für den angegebenen Zweck verarbeitet werden.

4. Zweckbindung: Daten dürfen nur für festgelegte Zwecke und welche, die mit den ursprünglich festgelegten Zweck vereinbar sind, verarbeitet werden.
5. Datensicherheit: Verantwortliche der Verarbeitung von Daten sind dazu verpflichtet, die Daten angemessen durch technische und organisatorische Maßnahmen zu schützen.

Zu den genannten Grundprinzipien werden zudem verschiedene Rechte von Bürgerinnen und Bürgern definiert, denen die Daten gehören. Dazu zählen das Recht auf Berichtigung, Löschung, Vergessenwerden und das Recht auf Einschränkung der Verarbeitung. Auch gibt es das Recht auf Datenübertragbarkeit, welches ermöglichen soll, dass Daten in einem gängigen maschinenlesbaren Format weitergegeben werden können, um etwa einen Wechsel eines Servicedienstleisters zu ermöglichen. Daneben gibt es noch das Widerspruchsrecht und das Recht auf Widerruf einer Einwilligung zur Verarbeitung. Schließlich wird das Recht auf nicht-automatisierte Entscheidungen definiert, wenn diese Entscheidungen rechtliche Auswirkungen haben und nicht rechtlich oder vertraglich eine Regelung dazu vorliegt, und das Recht auf Beschwerden gegenüber Aufsichtsbehörden, sollte ein Verstoß gegen das Datenschutzrecht vorliegen [7, S.57-63].

Diese Grundprinzipien zusammen mit den dazu definierten Rechten von Bürgerinnen und Bürgern führen dazu, dass die Verarbeitung von Daten beim maschinellen Lernen einigen Herausforderungen unterliegt. In der konkreten Umsetzung bedeuten diese Vorgaben, dass für die Verwendung von Daten beim maschinellen Lernen die betroffenen natürlichen Personen über die Verarbeitung der Daten im Sinne der Transparenz umfassend informiert werden müssen. Dies ist jedoch gerade beim maschinellen Lernen nicht immer einfach. Zwar ist der grundsätzliche Ablauf überschaubar, jedoch sind die inneren Abläufe der Modelle meist nicht verständlich, weshalb die Transparenz eine Anforderung ist, wofür nicht immer klare Lösungen existieren [40, S.15]. Auch gibt es das Recht auf Löschung der Daten, was in einem Szenario mit zentralen Daten zum maschinellen Lernen erhöhten Aufwand bedeutet, da diese Daten dann auch an zentraler Stelle entfernt werden müssen. Ein daraus trainiertes Modell darf zudem die Daten nicht herausgeben, was weiteren Schutz bei der Verarbeitung der Daten erfordert. Ein Schutz davor ist auch alleine schon nach dem Prinzip der Datensicherheit erforderlich.

Föderales Lernen dient hier als mögliche Lösung für einen Teil der Anforderungen beim Datenschutz, da es in seinem Grundkonzept bereits an den Datenschutz ansetzt, weil die Trainingsdaten nicht mehr an einem Ort liegen sondern verteilt und im Optimalfall liegen die Daten direkt bei den natürlichen Personen, denen sie gehören. Das Recht auf

Löschung, Einschränkung der Verarbeitung, das Widerspruchsrecht und das Recht zum Widerruf von Einwilligungen ist hierbei automatisch erfüllt, da die Daten nicht zentral gespeichert werden und bei den natürlichen Personen liegen, welche dadurch volle Kontrolle über die Verarbeitung haben. Auch das Recht auf Vergessenwerden, Berichtigung und Datenübertragbarkeit wird hierbei automatisch erfüllt, da ein zentraler Dienstleister gar nicht erst die Daten speichert. Damit diese Rechte vollständig erfüllt werden, muss allerdings durch weitere Maßnahmen sichergestellt werden, dass ein trainiertes Modell keine Daten herausgibt oder diese beim Trainingsprozess extrahiert werden können [40, S.17]. Das Recht auf nicht-automatisierte Entscheidungen kann durch das föderale Lernen allgemein nicht erfüllt werden, da wie zuvor erwähnt der Trainingsablauf von Modellen allgemein nicht transparent umsetzbar ist [40, S.16]. Die Datenminimierung im Sinne des Datenschutzes wird durch föderales Lernen erreicht, da ein zentraler Server gar nicht erst lokale Daten verarbeitet sondern nur die Aktualisierungen der Modelle aggregiert [40, S.15 f.]. Föderales Lernen schützt nicht davor, dass z.B. ein Modell Daten indirekt herausgibt oder aus den Aktualisierungen der Modelle Informationen extrahiert werden [40, S.2]. Auch im Sinne der Zweckbindung, Datensicherheit und Datenminimierung sollte verhindert werden, dass einzelne lokale Modelle bei der Aggregation oder aus dem globalen Modell ausgelesen werden können [40, S.15]. Hierzu und für die zuvor erwähnten Rechte sind weitere Maßnahmen erforderlich, die den Schutz der Daten und Modelle gewährleisten und die Daten gegen verschiedene Angriffszenarien auf Modelle schützen.

2.4 Schutzmechanismen beim föderalen Lernen

Beim föderalen Lernen können weitere Verfahren eingesetzt werden, um den Schutz der Daten zu erhöhen und damit gegen mögliche Angriffe vorzugehen. Eine Möglichkeit, die Daten besser zu schützen, ist diese zu anonymisieren, um den Informationsgehalt an personenbezogenen Daten zu reduzieren. Zwei der relevantesten Verfahren [40, S.3 f.], die weiter beim föderalen Lernen umgesetzt werden können, sind Differential Privacy und Secure Aggregation, die im Folgenden kurz erläutert werden.

2.4.1 Differential Privacy

Das Anonymisieren oder Entfernen von Teilen der Datensätze kann zwar den Schutz der Daten erhöhen, schützt diese jedoch nicht immer garantiert. Wie zuvor erwähnt können

z.B. bei einem Linkage Attack Daten deanonymisiert werden durch Kombination mit anderen Daten. Differential Privacy kann hierbei dazu genutzt werden, die Datensätze richtig zu schützen [10, S.6 f.]. Differential Privacy ist ein Verfahren, welches mathematisch sicherstellt, dass einzelne Einträge in einem Datensatz nur begrenzten Einfluss auf die Modellausgaben und Modellparameter haben und dadurch nur schwer aus diesen extrahiert werden können [10, S.20].

Differential Privacy schützt die Daten, indem den Daten ein Rauschen hinzugefügt wird und so sichergestellt wird, dass einzelne Einträge statistisch nicht unterscheidbar sind vom Rest der Daten, daher es keinen Unterschied für die Auswertung macht, ob ein Eintrag in den Trainingsdaten vorhanden war oder nicht. Dadurch kann auch nicht anhand der Verteilung auf Einträge in den Trainingsdaten geschlossen werden [10, S.6].

Ziel von Differential Privacy ist es, dass die Ausgaben eines Datensatzes D und eines Datensatzes D', welcher einen beliebigen Eintrag weniger hat (aber ansonsten die selben Einträge hat), nicht unterscheidbar sind. Genau genommen kann aus der Differenz der Datensätze D und D' keine Information über den Eintrag gewonnen werden. Differential Privacy wird zentral über den Parameter ϵ gesteuert, der angibt, wie stark das Rauschen ist. Die allgemeine Definition für ϵ-Differential Privacy lautet zunächst:

$$Pr[m_q(D) \in S] <= e^{\epsilon} * Pr[m_q(D') \in S] \tag{2.1}$$

q ist hierbei eine beliebige Funktion, die auf einem Datensatz angewendet wird, um eine statistische Auswertung durchzuführen. m_q ist q mit angewandtem Rauschen auf den Ausgaben und S ist eine Menge aller möglichen Ausgaben von m_q. Die Ungleichung formuliert somit, dass die Wahrscheinlichkeit, dass eine Ausgabe einer Funktion m_q auf einem Datensatz D in einer Menge S liegt, nicht mehr als e^{ϵ} mal so groß ist, wie die Wahrscheinlichkeit, dass die Ausgabe auf einem Datensatz D' in S liegt [20, S.17f].

Je kleiner ϵ gewählt wird, umso stärker wird das hinzugefügte Rauschen und die Daten werden besser geschützt. Zugleich sinkt mit kleineren ϵ die Genauigkeit des Modells, da die Daten immer mehr verfälscht werden und weniger Einfluss auf die Auswertung haben. Bei einem ϵ von 0 ist die Wahrscheinlichkeitsverteilung der Ausgaben von D und D' identisch und damit komplett geschützt, jedoch hat dies auch keine statistische Aussagekraft mehr, da der Datensatz nicht länger relevant für die Verteilung ist. Umso größer ϵ gewählt wird, umso weniger Rauschen wird hinzugefügt und einzelne Daten haben mehr Einfluss auf die Auswertung (die mögliche Differenz zwischen D und D'

wird größer). Die Genauigkeit des Modells steigt dadurch tendenziell, der Schutz der Daten sinkt jedoch [20, S.18].

ϵ-Differential Privacy begrenzt die Differenz rein durch ϵ. Hierbei ist es jedoch nicht einfach, einen Mechanismus zu entwerfen, der sowohl die Daten gut nach ϵ schützt als auch eine hohe Genauigkeit erzielt. Als Erweiterung dazu gibt es (ϵ, δ)-Differential Privacy, welches den Parameter δ einführt und lockerer ist als ϵ-Differential Privacy:

$$Pr[m_q(D) \in S] <= e^{\epsilon} * Pr[m_q(D') \in S] + \delta \tag{2.2}$$

Durch δ wird angegeben, zu welcher Wahrscheinlichkeit die ϵ-Differential Privacy verletzt werden darf [20, S.18]. Bei der Wahl des Wertes für δ sollte beachtet werden, dass dieser kleiner als $1/n$ ist (mit n als Anzahl der Einträge in der Datenbank), um die einzelne Einträge trotzdem möglichst gut zu schützen [10, S.18].

Differential Privacy bietet sowohl Vor- als auch Nachteile in der Anwendung. Wie bereits erwähnt, kann Differential Privacy den Schutz der Daten beim Training steuern, jedoch führt dies zu einem ungenaueren Modell. Zugleich ist Differential Privacy zukunfssicher, da die Sicherheit des Verfahrens mathematisch gegeben ist und die Daten können bei ausreichend gewählten Epsilon daher auch nicht durch z.B. weitere Datensätze über eine Linkage Attack zukünftig deanomyisiert werden [10, S.22 f.].

Beim föderalen Lernen kann Differential Privacy in zwei Varianten umgesetzt werden. In der allgemein bereits vorgestellten Variante, wird Differential Privacy auf das aggregierte Modell angewandt, welches verteilt wird, um das Herausgeben von Einträgen nach dem Training zu verhindern. In der lokalen Variante wird Differential Privacy bereits bei den Clients angewandt, um zu verhindern, dass über die Modellaktualisierungen Trainingsdaten extrahiert werden [23, S.57 f.].

2.4.2 Secure Aggregation

Secure Aggregation ist ein Verfahren zur Aggregation von Berechnungen, wobei die teilnehmenden Clients ihre Ergebnisse nicht direkt teilen. Es basiert auf der Secure Multi-Party Computation (SMPC), welche wiederum ursprünglich auf der Secure Computation basiert. Die Secure Computation definiert, wie zwei Parteien eine gemeinsame Berechnung durchführen können und dabei nicht die eigenen Werte der anderen Partei direkt übergeben müssen [44]. Die Secure Multi-Party Computation setzt dies für mehr als zwei

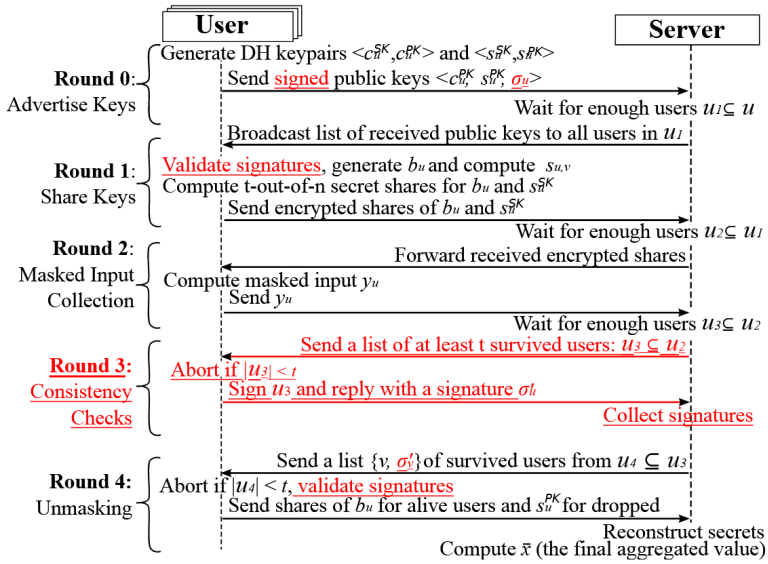

Abbildung 2.2: Ablauf der Secure Aggregation [5]

Parteien um [14]. Secure Aggregation definiert wiederum eine Untergruppe von Verfahren für Secure Multi-Party Computation, die Lösungen für die sichere Aggregation von Berechnungen ohne den direkten Austausch der Ergebnisse zwischen den Parteien bietet. Dies kann dazu genutzt werden, um beim föderalen Lernen die Aktualisierungen von Modellen bei einem zentralen Server zu aggregieren, ohne dass die beteiligten Clients oder der Server die einzelnen Aktualisierungen der Clients erfahren [5, S.1175].

Secure Multi-Party Computation schützt die Berechnungen allgemein, indem diese in einzelne Teile aufgeteilt werden und an unterschiedliche Clients verteilt werden. Einzelne Clients sammeln diese Teile und aggregieren sie und schicken das aggregierte Ergebnis an den zentralen Server. Dadurch erfährt der Server nicht, welche Aktualisierungen von welchem Clients stammen. Bei Secure Aggregation gilt das gleiche Prinzip, jedoch werden die Modellparameter, welche aggregiert werden sollen, nicht direkt aufgeteilt. Die Abbildung 2.2 stellt den Ablauf von Secure Aggregation in vereinfachter Form dar. Die Veröffentlichung dazu unterscheidet noch weiter zwischen zwei Modellen, die sich in den Sicherheitsgarantien unterscheiden, worauf hier jedoch nicht weiter eingegangen wird. Die roten Markierungen in der Abbildung stellen allerdings die Erweiterungen des zweiten Modells gegenüber dem ersten dar. Statt die Modellparameter aufzuteilen, werden diese mit individuellen Masken pro Client und Masken zwischen den Clients maskiert und direkt an den Server geschickt. Die Masken werden aufgeteilt und an die Clients geschickt, welche diese wiederum an den Server übermitteln. Die Masken sind so gewählt, dass

der Server die maskierten Modellparameter nicht einzeln demaskieren kann. Die Summe der Modellparameter kann nur durch die Aggregation aller Masken und maskierten Modellparameter bestimmt werden, wodurch der Server keinen Einblick in die einzelnen Modellparameter der Clients erhält. Der genauere Ablauf dazu, wie in Abbildung 2.2 auch vereinfacht dargestellt, sieht dabei wie folgend aus [5, S.1180 f.]:

- Runde 0: Alle Clients erzeugen jeweils 2 Schlüsselpaare und ggf. eine Signatur dazu. Die öffentlichen Schlüssel werden mit der Signatur an den Server geschickt und an die Clients verteilt.

- Runde 1: Die Clients validieren die öffentlichen Schlüssel anhand der Signaturen. Über das Diffie-Hellmann-Verfahren werden gemeinsame Schlüsselpaare erzeugt. Es werden Shares aus den privaten Schlüsseln der einen Schlüsselpaare nach Shamir's Secret Sharing erzeugt, wodurch ein privater Schlüssel in n Teile aufgeteilt wird, aber nur t Teile (t als gewählter Schwellwert) mindestens benötigt werden, um den privaten Schlüssel wiederherzustellen [5, S.1176]. Dies kann später dazu genutzt werden, die privaten Schlüssel wiederherzustellen. Auch werden Shares aus Zufallswerten als Seed für einen Zufallszahlengenerator erzeugt. Die Shares werden mit den gemeinsamen Schlüsselpaaren mit AES-GCM für die jeweiligen bestimmten Clients verschlüsselt. Der Server sammelt diese entstandenen Ciphertexte und verteilt sie an die Clients.

- Runde 2: Die Clients erhalten die für sie bestimmten verschlüsselten Shares. Aus den gemeinsamen Schlüsseln mit jeden Client und mit den eigenen Seeds werden jeweils Masken gebildet. Die Masken der gemeinsamen Schlüssel werden summiert zusammen mit der eigenen Maske auf die Modellparameter angewandt und die maskierten Modellparameter werden an den Server geschickt.

- (optionale) Runde 3: Um zu prüfen, welche Clients noch aktiv teilnehmen, erhalten alle Clients eine Liste der Clients, die noch teilnehmen, und bestätigen den Erhalt dieser durch eine Signatur der Liste dem Server. Der Server verteilt diese Signaturen an die Clients.

- Runde 4: Wenn Runde 3 durchgeführt wurde, prüfen alle Clients, ob die erhaltenen Signaturen der Liste der verbliebenen Clients stimmen. Alle Nutzer entschlüsseln zudem die zuvor erhaltenen Ciphertexte und erhalten daraus die Shares für die privaten Schlüssel und die Seeds. Der Server sammelt die entschlüsselten Shares der Clients und nutzt diese, um die Masken für alle Clients (sowohl aktive als auch

ausgefallene Clients) zu bestimmen. Der Server demaskiert darauf als Summe alle maskierten Modellparameter mithilfe der Masken und erhält dadurch die summierten Modellparameter.

In der ersten Veröffentlichung zu Secure Aggregation nennen Bonawitz et al. einige Vorteile der Secure Aggregation. Die Anzahl der benötigten Kommunikationsrunden für die Secure Aggregation ist konstant und die Secure Aggregation führt nur zu wenig Mehraufwand bei der Kommunikation. Auch bei Fehlern (ein Client sendet z.B. seinen Wert nicht) funktioniert die Secure Aggregation weiterhin und benötigt zudem nur einen Server, um die Kommunikation zu steuern [5, S.1176].

2.5 Techniken zur Textdarstellung

Im Anwendungsfall dieser Arbeit werden Texte klassifiziert, wofür zunächst ein Verfahren benötigt wird, um die Texte in eine Form zu bringen, welche für die weitere Klassifizierung genutzt werden kann.

2.5.1 Word2Vec

Word2Vec ist eine Technik, um aus Wörtern Vektoren zu erzeugen, die diese Worte repräsentieren. Mit dieser Technik können Modelle trainiert werden, die auf großen Datensätzen basieren und ein Vokabular von Millionen von Wörtern haben können (z.B. hat Google ein Modell basierend auf Google News Daten mit über 100 Milliarden Wörtern trainiert und einem Vokabular von über 3 Millionen Wörtern [16]). Hierbei werden Wörter im Vektorraum so dargestellt, dass Wörter mit einer ähnlichen Bedeutung bzw. Verwendung in ähnlichen Kontexten einen ähnlichen Vektor haben in einzelnen Dimensionen. So kann ein Wortvektor gleichzeitig auch ein Wort in verschiedenen Kontexten darstellen. Gleichzeitig drücken die Distanzen zwischen den Wortvektoren auch Beziehungen zwischen den Wörtern aus. Ein Beispiel, was dazu oft genannt wird, ist, dass die Rechnung Vektor König - Vektor Mann + Vektor Frau den Vektor für Königin in einen trainierten Modell ergeben könnte [26, S.1f]. Bei der Wahl der Parameter für Word2Vec ist vor allem die Fenstergröße wichtig und die Größe der Wortvektoren. Die Fenstergröße bestimmt, wie viele Kontextwörter um ein Wort herum betrachtet werden. Für die Umsetzung von Word2Vec gibt es zwei verschiedene Architekturen: Continuous Bag-of-Words (CBOW) und Skip-gram [26, S.4].

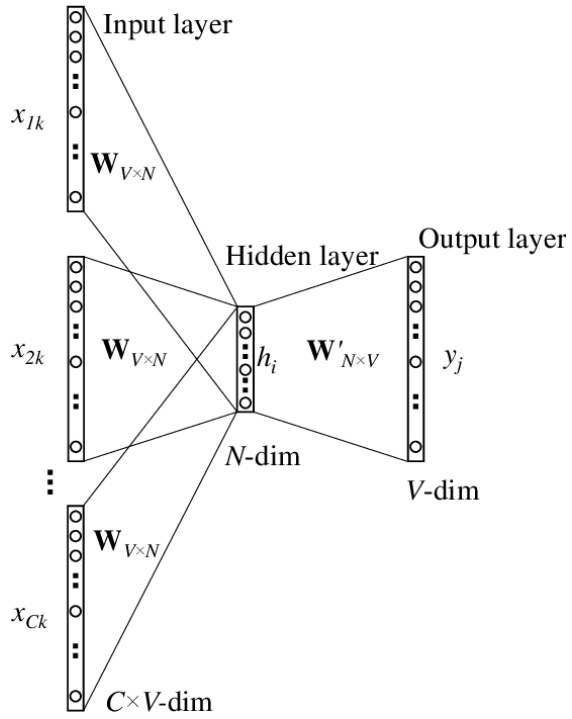

Abbildung 2.3: CBOW-Architektur von Word2Vec [32]

Das Continuous Bag-of-Words (CBOW) Modell versucht anhand des Kontextes von Wörtern in einem Text ein Wort vorherzusagen. Die Abbildung 2.3 stellt die Architektur von CBOW dar. Dazu werden sowohl die vergangenen Wörter vor dem Wort als auch die Wörter nach dem Wort betrachtet. Das CBOW-Modell erhält in der Eingabeschicht die Kontextwörter (Anzahl C) als One-Hot-Kodierung. Eine One-Hot-Kodierung stellt Wörter als Vektor mit der Größe des Vokabulars (V) dar. Das jeweilige Wort wird dabei als 1 dargestellt und alle anderen Wörter als 0 [32, S.1f]. Aus den One-Hot-Kodierungen der Kontextwörter in der Eingabeschicht wird der Durchschnitt der Vektoren gebildet [32, S.5], um die Ausgabewerte für die Hidden-Schicht zu bestimmen. Die Ausgabeschicht hat schließlich so viele Knoten wie die Größe des Vokabulars und gibt damit als One-Hot-Kodierung das Wort zurück, welches vorhergesagt werden soll. Die Wortvektoren ergeben sich aus den Gewichten zwischen der Eingabe- und der Hidden-Schicht, wobei jede Zeile der Gewichtsmatrix den Wortvektor für ein Wort darstellt. Die Größe der Hidden-Schicht (N) bestimmt wiederum, wie lang die Wortvektoren sind [32, S.1f].

Das Skip-gram-Modell versucht hingegen, anhand eines Wortes den Kontext des Wortes vorherzusagen. Die Abbildung 2.4 stellt die Architektur von Skip-gram dar. Somit wird

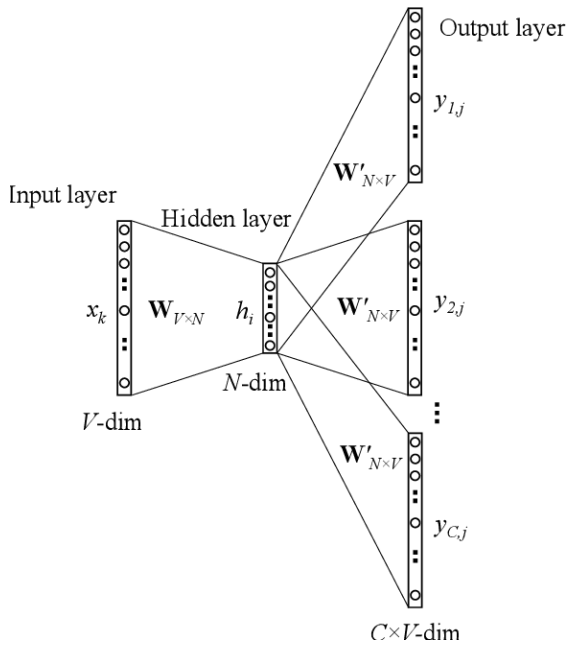

Abbildung 2.4: Skip-gram-Architektur von Word2Vec [32]

der Eingabeschicht die One-Hot-Kodierung des Wortes übergeben. Die Ausgabeschicht gibt wiederum die One-Hot-Kodierungen der Kontextwörter zurück [32, S.7f].

Welche der beiden Architekturen verwendet werden sollte, hängt von den Anforderungen ab. Zum einen variieren beide Modelle in der benötigten Trainingszeit. Skip-gram benötigt im Allgemeinen mehr Zeit zum Trainieren als CBOW, da Skip-gram mehr Ausgaben hat, auf die ein Modell optimiert wird. Zudem ist Skip-gram besser darin, semantische Bedeutungen zu erfassen, während CBOW besser darin ist, syntaktische Strukturen zu erfassen [26, S.7-9].

2.6 Klassifikationsverfahren

Für die Umsetzung des Anwendungsfalls dieser Arbeit werden Klassifikationsverfahren benötigt, welche genutzt werden können, um Texte zu klassifizieren. Im Folgenden wird eine Auswahl an Klassifikationsverfahren erläutert, die für die Klassifizierung von Texten geeignet sind.

2.6.1 Naive Bayes

Naive Bayes sind eine Gruppe von Klassifikatoren, die auf dem Satz von Bayes basieren. Der Naive Bayes ermöglicht die Bestimmung der Wahrscheinlichkeit, dass eine Eingabe zu einer bestimmten Klasse gehört. Genauer kann mit dem Naive Bayes bestimmt werden, wie wahrscheinlich es ist, dass bei gegebenen Beobachtungen eine Hypothese wahr ist (bedingte Wahrscheinlichkeit). Der Satz des Bayes, auf dem der Naive Bayes basiert, besagt, dass mit einer Hypothese H und den Wahrscheinlichkeiten, dass eine Beobachtung E eintritt, wenn H wahr ist, die Wahrscheinlichkeit, dass H wahr ist, wenn E eintritt, bestimmt werden kann:

$$Pr[H|E] = \frac{Pr[E|H] * Pr[H]}{Pr[E]} \tag{2.3}$$

Der Naive Bayes gilt als naiv, da davon ausgegangen wird, dass die Beobachtungen, die E ausmachen, alle unabhängig voneinander sind, was in der Realität meist nicht der Fall ist. Dadurch lässt sich $Pr[E|H]$ aus den Wahrscheinlichkeiten der einzelnen Beobachtungen berechnen (mit n als Anzahl der möglichen Beobachtungen):

$$Pr[E|H] = Pr[E_1|H] * Pr[E_2|H] * ... * Pr[E_n|H] \tag{2.4}$$

Hierbei besteht jedoch ein Problem, da die Wahrscheinlichkeit $Pr[E|H] = 0$ ist, wenn eine einzelne Beobachtung E_n nicht vorkommt, wodurch insgesamt die Wahrscheinlichkeit für eine Hypothese bei gegebenen Beobachtungen $Pr[H|E] = 0$ ist. Dadurch würden alle Beobachtungen ignoriert werden, sobald eine nicht vorhanden ist (etwa das Vorkommen eines Wortes in einem Text). Um dies zu verhindern wird die Laplace-Glättung angewandt, indem das Auftreten jeder Beobachtung um eine bestimmte Konstante erhöht wird (und die gesamte Anzahl der Beobachtungen). Die Wahrscheinlichkeit für eine einzelne Beobachtung $Pr[E_i|H]$ ergibt sich somit aus $Pr[E_i|H] = \frac{N_i+a}{N+a*n}$, wobei N_i die Anzahl der Vorkommen der Beobachtung E_i in H ist, N die Anzahl aller Vorkommen aller Beobachtungen in H, a die Konstante zur Laplace-Glättung und n die Anzahl der möglichen Beobachtungen ist [1, S.90 f.].

Zum Naive Bayes gibt es verschiedene Varianten von Klassifikatoren die für verschiedene Anwendungsfälle gut geeignet sind. Die bekanntesten sind der Gaussian Naive Bayes, der Multinomial Naive Bayes und der Bernoulli Naive Bayes. Betrachtet wird hier nur der Multinomial Naive Bayes, welcher am häufigsten in der Klassifizierung für diskrete Daten verwendet wird, da dieser für Textklassifizierung am besten geeignet ist. Mit dem Multinomial Naive Bayes können die Häufigkeiten aller Wörter beachtet werden, da die

Verteilung der Wörter in einem Dokument als Multinomialverteilung betrachtet wird. Somit wird nicht nur das binäre Vorkommen eines Wortes in einer Klasse beachtet, sondern auch die Häufigkeit des Wortes in der Klasse. Typischerweise wird dabei ein Dokument als Bag of Words betrachtet, wo die Anzahl der Wörter in einem Dokument gezählt wird [1, S.94-96].

2.6.2 Random Forest

Random Forest ist ein Verfahren, welches auf dem Konzept von Entscheidungsbäumen aufbaut. Entscheidungsbäume ermöglichen eine Klassifizierung durch eine Aufteilung in viele Entscheidungen basierend auf den Attributen von Daten. Schonlau und Zou beschreiben die Bildung eines Entscheidungsbaums grundlegend als das Aufteilen eines Datensatzes nach gegebenen Kriterien in je zwei Gruppen, bis eine vorher festgelegte Abbruchbedingung erreicht wird. Am Ende eines Entscheidungsbaumes befindet sich dann ein Blattknoten/Blatt, welcher eine Entscheidung darstellt. Bei der Bildung eines Entscheidungsbaumes, wird an jedem Knoten ein Attribut anhand eines zuvor festgeleten Kriteriums ausgewählt. Dieses Kriterium kann zum Beispiel die Entropie oder der Gini-Index sein. Die Knoten werden anschließend solange aufgeteilt, bis eine der Abbruchbedingungen erreicht wird, wie etwa eine maximale Tiefe des Baumes oder eine klare Trennung der Daten in den Blattknoten nach gegebenen Kriterium [35, S.4 f.].

Problem bei einfachen Entscheidungsbäumen ist jedoch, dass diese sehr anfällig für Overfitting sind und daher schnell bei neuen Daten schlechtere Ergebnisse liefern können. Hier kommt Random Forest als Verfahren zur Lösung dieses Problems zum Einsatz: Bei Random Forest werden viele unterschiedliche Entscheidungsbäume erstellt und befragt und das Ergebnis dieser Bäume aggregiert. Die Bäume werden erstellt, indem zunächst Bootstrap-Daten aus den Trainingsdaten entnommen werden. Bootstrap-Daten entstehen, indem zufällig Daten aus den Trainingsdaten ausgewählt werden, bis die Größe der Trainingsdaten erreicht ist. Dabei dürfen auch Daten mehrfach in den Bootstrap-Datensatz aufgenommen werden. Für jeden neuen Entscheidungsbaum wird ein neuer Bootstrap-Datensatz erstellt. Zum Aufbau der Entscheidungsbäume werden n zufällige Attribute aus den Daten ausgewählt und zwischen diesen wird sich nach einem Kriterium für das beste Attribut entschieden. Bei der Evaluation werden dann in einem Mehrheitsentscheid alle Bäume im Wald befragt. Der Prozess wird auch Bagging genannt, da mit Bootstrap-Daten gearbeitet wird und aggregiert wird. Bei der Bildung der Bootstrap-Daten werden etwa ein Drittel der Trainingsdaten nicht in den Bootstrap-Datensatz

aufgenommen. Diese Daten werden als Out-of-Bag-Daten bezeichnet und können zur Validierung der Bäume genutzt werden. Als Out-of-Bag-Error kann der Anteil der falsch klassifizierten Out-of-Bag-Daten berechnet werden. Zur Verbesserung des Random Forests können dann diverse Parameter wie die Anzahl der zufällig ausgewählten Attribute oder die Baumtiefe angepasst und die Veränderung des Out-of-Bag-Errors beobachtet werden [35, S.5 f.].

2.6.3 Long short-term memory

Long short-term memory (LSTM) ist eine Technik basierend auf künstlichen neuronalen Netzen. LSTM-Modelle sind besonders gut in der Verarbeitung von Sequenzen und Zeitreihen, da sie mit einem Kurz- und Langzeitgedächtnis arbeiten und grundsätzlich eine beliebige Anzahl an Eingaben verarbeiten können. So können LSTM-Modelle auch Zusammenhänge über längere Zeit/längere Sequenzen erkennen, da relevante Informationen nicht so leicht in Vergessenheit geraten. Dadurch eignen sie sich auch zur Klassifikation von Texten, da Wort für Wort Texte durch LSTM-Modelle verarbeitet werden können und der gesamte Zusammenhang beachtet wird [15, S.410 f.].

Ein LSTM-Modell ähnelt den rekurrenten neuronalen Netzen (RNN). Ein RNN ermöglicht Verbindungen zur gleichen Schicht im Netz oder zu vorherigen Schichten, wodurch Eingaben beliebiger Länge verarbeitet werden können mit dem selben Modell und unter Verwendung der gleichen Gewichte, weshalb die Größe des Modells nicht mit wachsender Größe der Eingabe ansteigt [15, S.373 f.]. Ein Problem ist jedoch, dass einfache RNNs allgemein immer schlechter Informationen auf lange Zeit verarbeiten können, da hierbei das Vanishing/Exploding Gradient Problem auftritt. Durch wiederholte Multiplikationen mit den gleichen Gewichten werden die Werte des neuronalen Netzes immer größer oder immer kleiner, wodurch auch die Gradienten immer größer oder immer kleiner werden und nicht klar wird beim Training, in welche Richtung die Gewichte angepasst werden sollen [15, S.289 f.].

LSTM-Modelle lösen dieses Problem. In der Abbildung 2.5 wird ein LSTM-Block dargestellt, welcher als Teil eines Modells verwendet wird. Wie bei den einfachen RNNs erhalten LSTM-Modelle neben der Eingabe weitere Informationen als Vektoren aus vorherigen Schritten im Modell. Diese Vektoren sind das „Hidden State“ (Kurzzeitgedächtnis) und „Cell State“ (Langzeitgedächtnis), welche vom vorherigen LSTM-Block zurückgegeben wurden (oder Initialvektoren beim ersten Durchlauf). Daneben geben die LSTM-Blöcke

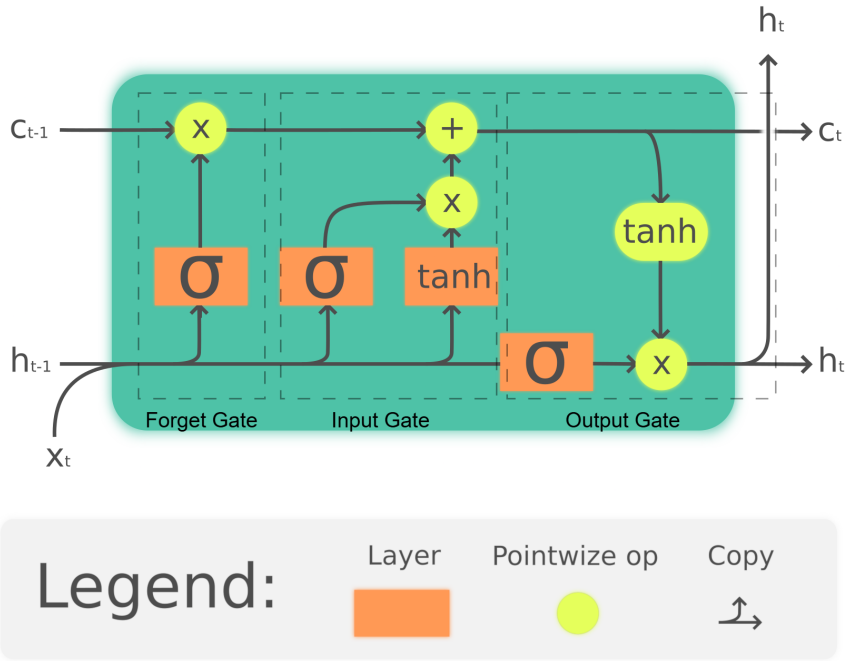

Abbildung 2.5: Vereinfachte Darstellung eines LSTM-Modells [8]

ihr jeweiliges Hidden State insgesamt für die Ausgabe zurück. Ein LSTM-Block besteht dabei aus mehreren Schichten. Zentral durch einen LSTM-Block verläuft das Cell State. Daneben verläuft das Hidden State, von wo aus über „Gates“ das Langzeitgedächtnis verändert wird. Ein LSTM-Block besteht aus 3 Gates, das „Forget Gate“, das „Input Gate“ und das „Output Gate“ [29]:

- Forget Gate: Das Forget Gate entscheidet, welcher Anteil der Informationen im Langzeitgedächtnis behalten wird.
- Input Gate: Das Input Gate kombiniert das Kurzzeitgedächtnis mit der aktuellen Eingabe und bestimmt, welcher Anteil dieser Informationen ins Langzeitgedächtnis übergeht.
- Output Gate: Das Output Gate dient dazu, die Informationen im Kurzzeitgedächtnis zu aktualisieren. Mithilfe des vorherigen Kurzzeitgedächtnisses und der aktuellen Eingabe wird bestimmt, welcher Anteil des neuen Langzeitgedächtnisses im Kurzzeitgedächtnis bleibt und als Ausgabe des LSTM-Blocks dient.

3 Anwendungsfall

Die grundlegenden Konzepte der Arbeit sind nun bekannt. Im folgenden Kapitel wird der spezifische Anwendungsfall genauer erläutert und dessen Eignung für das föderale Lernen erklärt, welcher in der späteren Implementierung umgesetzt wird und auf dessen Grundlage der Vergleich von föderalem und zentralem Lernen durchgeführt wird.

Der zentrale Anwendungsfall der Arbeit ist die Kategorisierung von E-Mail-Nachrichten. Da die Trainingsdaten dabei bereits gelabelt vorliegen, werden dazu Verfahren des überwachten Lernens angewandt. Die Kategorisierung von E-Mail-Nachrichten ist ein bereits weit verbreitetes Gebiet im Bereich der Textklassifizierung, wo auch in der praktischen Umsetzung verschiedene Verfahren des maschinellen Lernens genutzt werden. Dadurch eignet sich die Kategorisierung von E-Mail-Nachrichten als Anwendungsfall, da so der Fokus mehr auf das föderale Lernen gelegt werden kann. Für den Anwendungsfall wird davon ausgegangen, dass es sich bei den Clients um Behörden wie etwa verschiedene Finanzämter handelt. Da es sich bei den Clients um Finanzämter handelt, ist es besonders wichtig, dass die Nachrichten möglichst geschützt sind bei der Kategorisierung, da sie personenbezogene Daten enthalten könnten und daher aus Sicht des Datenschutzes besonders schützenswert sind. Die Anzahl der Finanzämter, die dabei einbezogen werden, ist eher gering im Vergleich zu anderen klassischen Anwendungsfällen mit z.B. mobilen Endgeräten (in Hamburg gibt es z.B. 10 Regionalfinanzämter und 4 Sonderfinanzämter), weshalb nur eine geringe Anzahl an Clients am föderalen Lernen teilnimmt. Die einzelnen Finanzämter erhalten und versenden auch nicht unbedingt gleich viele Nachrichten. Die Arten der Nachrichten sind dabei bei allen Finanzämtern erst einmal grundsätzlich ähnlich, können sich jedoch auch stärker unterscheiden, wenn z.B. spezielle Zuständigkeiten für bestimmte Bereiche vorhanden sind. Bei der Kategorisierung soll grundlegend erst einmal zwischen Spam und Nicht-Spam-Nachrichten unterschieden werden und nach Möglichkeit kann auch eine weitere Unterteilung in verschiedene Kategorien erfolgen. Die Datengrundlage für die Umsetzung des Anwendungsfalls basiert allerdings nicht auf Mails von Finanzämtern, da diese zum Zeitpunkt der Umsetzung nicht verfügbar waren.

Der Anwendungsfall kann aufgrund der Eigenschaften als Cross-Silo federated learning betrachtet werden, wie in Abschnitt 2.2.1 beschrieben. Die Daten liegen dabei in klassisch horizontaler Verteilung vor, da jeder Client verschiedene Mails als Trainingsdaten nutzt, jedoch alle gleich klassifiziert werden. Für Cross-Silo federated learning spricht, dass es sich bei den Finanzämtern nur um wenige Clients handelt, deren Daten geschützt werden sollen beim Training. Genauer handelt es sich bei den Clients um feste Server, die möglichst immer verfügbar und eher leistungsstark sind im Vergleich zum klassischen Szenario beim föderalen Lernen mit mobilen Endgeräten. Die einzelnen Clients haben jeweils ein hohes Datenaufkommen, da sie viele E-Mail-Nachrichten erhalten und versenden. Die Nachrichten aller Clients sind grundsätzlich erst einmal relevant, da sie unterschiedliche Nachrichten erhalten und versenden. So könnte z.B. ein Finanzamt bestimmte Spamnachrichten bekommen, die ein anderes (noch) nicht erhält.

4 Verwandte Arbeiten/Projekte

Das Ziel dieser Arbeit und der konkrete Anwendungsfall, welcher in der späteren Implementierung umgesetzt wird, wurden bereits erläutert. Im folgenden Kapitel werden nun noch verwandte Arbeiten und Projekte vorgestellt, welche sich auch mit dem föderalen Lernen auseinandersetzen und ähnliche Fragestellungen/Probleme behandeln oder auch föderales Lernen auf einen ähnlichen Anwendungsfall anwenden. Dazu werden die einzelnen Arbeiten kurz vorgestellt und mit dem Ziel und der Umsetzung dieser Arbeit verglichen.

Die Arbeit „Comparison of Deep and Traditional Learning Methods for Email Spam Filtering“ von Abdullah Sheneamer [36] beschäftigt sich mit den Vergleich von verschiedenen Methoden zur Spamklassifizierung von E-Mails und SMS im Bereich des maschinellen Lernens. Für den Vergleich werden Support Vector Machines (SVM), Gaussian Naive Bayes, Entscheidungsbäume, Random Forest, Xgboost, sowie LSTM- und CNN-Modelle verwendet. Für das LSTM- und CNN-Modell wurde zudem noch probiert, für die Word Embeddings GLOVE zu verwenden. Die Modelle werden anhand der Metriken Accuracy, Precision, Recall und F1-Score verglichen. Die Methoden werden anhand dreier Enron-Datensätze und einem SMS-Spam-Datensatz getestet. Der Enron-Datensatz wird auch in dieser Arbeit als Datengrundlage verwendet, wie später noch genauer erläutert wird. Im Vergleich zu dieser Arbeit konzentriert sich Sheneamers Arbeit auf den Vergleich der Methoden bei der Spamklassifizierung von E-Mails (und SMS), während diese Arbeit föderales Lernen im Fokus hat und die Methoden nur dazu dienen, föderales Lernen selbst und im Vergleich zum zentralen Lernen zu evaluieren. Auch vergleicht Sheneamer ein breiteres Spektrum an Methoden, während hier nur mit Naive Bayes, Random Forest und LSTM drei wesentliche Methoden verglichen werden. Anzumerken ist dazu noch, dass Sheneamer für die Word Embeddings für das LSTM-Modell GLOVE verwendet, während hier Word2Vec verwendet wird.

Die Veröffentlichung „Communication-Efficient Learning of Deep Networks from Decentralized Data“ von McMahan et al. [24] untersucht neben der eigentlichen Vorstellung

des föderalen Lernens mit Federated Averaging auch das vorgestellte Verfahren anhand zweier Anwendungsfälle. Hier wird Federated Averaging bei der Bildklassifizierung und bei der Texterzeugung mit einem Large Language Model angewendet. In beiden Fällen wird eine hohe Anzahl Clients im Vergleich zu dieser Arbeit betrachtet mit 100 und 1146 Clients. Auch wird hier nicht weiter die Anwendung weiterer Schutzmechanismen zum Schutz der Daten betrachtet. Hingegen werden hier, wie auch in der Veröffentlichung vorgeschlagen, Differential Privacy und Secure Aggregation als weitere Verfahren genutzt. Zudem wird in dieser Arbeit ein direkter Vergleich zum zentralen Lernen mit dem föderalen Lernen durchgeführt.

Am ähnlichsten zu dieser Arbeit ist die Veröffentlichung „Spam Email Categorization with NLP and Using Federated Deep Learning“ von Ul Haq et al [41], welche sich auch mit der Kategorisierung von E-Mails beim föderalen Lernen beschäftigt. Die Veröffentlichung legt den Fokus darauf, „Federated Phishing Filtering“ zu diesem Zweck kurz vorzustellen, um SPAM, HAM und Phising-Mails zu klassifizieren und ein Modell zu trainieren, welches diese in Echtzeit erkennt und mit aktuellen Daten aktualisiert werden kann. Hierbei wird jeder E-Mail-Nutzer als ein Client im föderalen Lernen betrachtet. Zur Darstellung der Texte wird TF-IDF eingesetzt und es wird auch Differential Privacy zum weiteren Schutz der Daten verwendet. Zur Kombination der Modelle der Clients zu einem globalen Modell wird Federated Averaging grundsätzlich verwendet, wobei eine Variante mit Gewichtung nach der Größe der Datensätze der Clients und eine Variante ohne untersucht wurden. Als Datengrundlage wurden Datensätze von CEAS2008, Spam Assassin, Jose Phishing, und Untroubled.org verwendet. Evaluiert wurden die Ansätze anhand der Genauigkeit der Modelle. Um bei der Evaluation zu demonstrieren, wie gut das Vorgehen bei sich verändernder Umgebung funktioniert, wenn etwa neue Arten von SPAM-Mails hinzukommen, wurde zunächst auf Grundlage einer Gruppe von Datensätzen trainiert, um dann in weiteren Trainingsverlauf weitere Daten der anderen Datensätze hinzuzunehmen. Die Veröffentlichung geht nicht genauer darauf ein, welches Verfahren zur Klassifizierung der Mails verwendet wurde. In dieser Arbeit findet hingegen ein breiterer Vergleich von Verfahren zur Klassifizierung beim föderalen Lernen statt mit verschiedenen Szenarien und es wird direkter ein Vergleich mit dem zentralen Lernen durchgeführt. Allerdings wird keine sich verändernde Umgebung simuliert, um die Anpassungsfähigkeit in den verschiedenen Szenarien zu testen.

Im Kontext der Arbeit ist auch das Projekt „TRUMPET“ relevant. Im Rahmen des „Horizon Europe“-Programms der EU [11] beschäftigt sich das Projekt „TRUstworthy Multi-site Privacy Enhancing Technologies“ (TRUMPET) [31] mit dem Einsatz von fö-

deralem Lernen im Gesundheitswesen. Zentrales Ziel des Projekts ist es, eine Plattform zu entwickeln, die es ermöglichen soll, Pantientendaten von Krankenhäusern oder Forschungseinrichtungen gemeinsam zu analysieren und zwischen diesen vergleichen zu können, ohne dabei den Datenschutz nach der DSGVO zu verletzen. Zentraler Bestandteil des Projekts ist es, föderales Lernen mit weiteren Schutzmechanismen zur Verbesserung der Privatsphäre einzusetzen. Dazu gehören zentral im Projekt Homomorphic Encryption, Secure Multi-Party Computation, Differential Privacy und Coded Distributed Computing, welche zum selben Zweck zum Teil auch in dieser Arbeit eingesetzt werden. Neben der Entwicklung der Plattform gehören zu den Zielen des Projekts, Gefahren zum Datenschutz zu untersuchen und Metriken zu schaffen, um den Schutz der Daten bewerten zu können. Zudem ist es Ziel des Projekts, allgemein zu untersuchen, ob die Plattform die Daten rechtlich ausreichend schützt, und weitere Nutzungsmöglichkeiten der Plattform zu untersuchen. Das Projekt läuft noch bis September 2025 und Ergebnisse im Kontext des Projekts werden über verschiedene Veröffentlichungen und Veranstaltungen fortlaufend bekannt gegeben.

Ein ähnliches Projekt unter dem „Horizon Europe“-Programm ist „Machine learning to augment shared knowledge in federated privacy-preserving scenarios“ (MUSKETEER) [27]. Das Projekt beschäftigt sich ebenfalls mit dem Datenschutz beim föderalen Lernen und hat als Ziel, eine allgemeine Plattform zu entwickeln, die es ermöglicht, mit föderalem Lernen Modelle zu trainieren und die Daten der Clients dabei zu schützen. Dazu wurden für zwei Anwendungsfälle verschiedene Modelle trainiert in einer dafür entwickelten Architektur und dabei verschiedene Methoden probiert, um die Daten zu schützen. Die Ergebnisse des Projekts sind öffentlich zugänglich, wie auch die dazu entwickelte Bibliothek zum Training der Modelle [28].

5 Eingesetzte Techniken und Verfahren

Zuvor wurden bereits in den Grundlagen verschiedene Techniken und Verfahren vorgestellt, die im Kontext der Arbeit relevant sind. Im folgenden Kapitel wird nun erläutert, welche konkreten Techniken und Verfahren für die Umsetzung des Anwendungsfalls eingesetzt werden und warum.

Zunächst wird auf die verwendeten Techniken eingegangen, die unabhängig des Kontextes im föderalen Lernen dazu genutzt werden, um die Klassifizierung von E-Mails als Anwendungsfall durchzuführen. Hierbei wird aus Sicht eines zentral trainierten Modells ohne föderales Lernen gedacht, da für den Vergleich die jeweilige föderale Umsetzung auf der zentralen Variante basiert. Als Trainingsdaten dienen verschiedene E-Mails, auf die im späteren Verlauf noch genauer eingegangen wird. Diese E-Mails bestehen aus Texten, welche zunächst numerisch kodiert werden müssen, damit sie anschließend klassifiziert werden können. Zur Kodierung von Worten und Texten gibt es verschiedene Methoden, von denen Word2Vec ausgewählt wurde. Als Alternative zur Kodierung der Wörter wurde auch TF-IDF betrachtet. Word2Vec hat gegenüber anderen Methoden wie TF-IDF jedoch den Vorteil, dass Word2Vec die semantische Bedeutung von Wörtern erfasst und danach auch Wörter in der Repräsentation unterscheidet, was eine genauere Klassifikation ermöglicht. Auch die Bedeutung von Wörtern im Kontext einer E-Mail wird durch Word2Vec berücksichtigt und die Wortvektoren können direkt als Eingabe für z.B. LSTM-Modelle verwendet werden, da diese eine Vektordarstellung als Eingabe benötigen.

Das gebildete Word Embedding durch Word2Vec wird anschließend an ein Klassifikationsverfahren übergeben, welches die E-Mails in Kategorien einteilen soll. Dazu werden in dieser Arbeit mehrere Klassifikationsverfahren für den Vergleich zwischen zentralem und föderalem Lernen verwendet, um den Vergleich weniger von dem eigentlichen Klassifikationsverfahren abhängig zu machen und auch mehr Unterschiede zu betrachten, die eventuell nicht bei allen Klassifikationsverfahren auffallen. Bei der Wahl der Klassifikationsverfahren wurden typische Verfahren zur Textklassifizierung mit unterschiedlicher

Effektivität betrachtet. Die Wahl wurde primär anhand der Veröffentlichung „Comparison of Deep and Traditional Learning Methods for Email Spam Filtering“ von Abdullah Sheneamer [36] getroffen, welche verschiedene Techniken zur Klassifizierung von E-Mails in Spam und Ham vergleicht und damit eine gute Grundlage für die Wahl der Verfahren bietet. Sheneamer nutzt dafür den Enron-Datensatz, welcher auch in dieser Arbeit für die Ham-Mails verwendet wird, und zudem noch einen SMS-Spam-Datensatz. Die Tabelle 5.1 fasst die Ergebnisse der Veröffentlichung zusammen (ein genauerer Überblick über die Metriken ist in der Veröffentlichung zu finden). Die Metriken wurden dazu als F1-Score zusammengefasst und die Tabelle stellt die Ergebnisse für die einzelnen verwendeten Datensätze der Veröffentlichung dar.

Tabelle 5.1: Zusammenfassung (als F1-Score) der Ergebnisse von „Comparison of Deep and Traditional Learning Methods for Email Spam Filtering“ [36]

Verfahren	Enron1	Enron2	Enron3	SMSSpamCollection	Alle
SVM	86,03%	81,66%	86,24%	88,68%	86,03%
Naive Bayes (GaussianNB)	80,88%	77,50%	82,55%	84,55%	80,88%
Decision Tree	79,55%	81,70%	81,11%	88,40%	79,37%
Random Forest	88,19%	88,40%	89,01%	90,57%	88,12%
Xgboost	87,16%	86,94%	88,48%	91,35%	87,16%
LSTM	98,11%	98,96%	98,67%	99,22%	97,39%
LSTM mit GLOVE	95,76%	96,07%	96,05%	98,80%	97,18%
CNN	97,67%	98,60%	98,62%	99,04%	97,34%
CNN mit GLOVE	97,66%	98,50%	98,30%	99,13%	97,51%

Für diese Arbeit wurde zum einen der Gaussian Naive Bayes zur Klassifizierung ausgewählt, welcher weit verbreitet in der Klassifizierung von Texten und vor allem bei der Spamerkennung verwendet wird. Aufgrund der weiten Verbreitung des Gaussian Naive Bayes und weil es sich um ein eher simples Verfahren zur Klassifizierung handelt, wird dieser als erstes Verfahren in dieser Arbeit verwendet. Dabei ist zu beachten, dass föderales Lernen sich derzeit noch vor allem auf die Umsetzung mit neuronalen Netzen konzentriert und eher wenig umgesetzte Lösungen für klassische Klassifikationsverfahren existieren. Dies liegt vermutlich daran, dass föderales Lernen ursprünglich für neuronale Netze präsentiert wurde und neuronale Netze auf Parametern basieren, die leichter zu aggregieren sind als etwa Random Forests. Die allgemeine Beliebtheit von neuronalen Netzen aufgrund der breiten Einsatzmöglichkeiten spielt dabei vermutlich auch eine Rolle.

Sheneamer vergleicht auch das Random Forest Verfahren, welches als weiteres Verfahren für den Vergleich in dieser Arbeit genutzt wird, da es nach der Veröffentlichung von Sheneamer bessere Ergebnisse als der Gaussian Naive Bayes erzielt und auch ein weit verbreitetes und einfaches Verfahren ist. Zudem erscheint die Verwendung von Random Forest im Kontext des föderalen Lernens zunächst als gut geeignet, da beim Random Forest verschiedene Entscheidungsbäume trainiert werden, um eine gemeinsame Entscheidung zu treffen. Ein Random Forest könnte somit beim föderalen Lernen aus Entscheidungsbäumen von verschiedenen Clients bestehen, um anschließend eine gemeinsame Entscheidung zu treffen.

Schließlich wurden LSTMs als weiteres Verfahren für die Klassifizierung in dieser Arbeit ausgewählt, welches in der Veröffentlichung von Sheneamer die besten Ergebnisse erzielt hat. Zudem handelt es sich um einen Ansatz mit neuronalen Netzen, wodurch sich dieser besonders gut für die Umsetzung eignet, da der Bereich des föderalen Lernens sich derzeit primär auf die Umsetzung mit neuronalen Netzen konzentriert. Dadurch gibt es bereits zahlreiche Implementierungen, die für LSTMs genutzt werden können und auch Verfahren wie FedAvg können direkt darauf angewandt werden. Außerdem sind LSTMs auch ein weit verbreitetes Verfahren, was auch im Bereich der Textklassifizierung viel genutzt wird.

Um den Schutz der Daten beim föderalen Lernen im Anwendungsfall zu gewährleisten, werden zudem Differential Privacy und Secure Aggregation eingesetzt. Secure Aggregation verhindert Reconstruction Attacks, da Angreifer, die etwa die Kommunikation abfangen oder direkt auf den Server Zugriff haben, nicht die einzelnen Modellaktualisierungen der Clients einsehen können, weil diese nur aggregiert insgesamt von allen Clients eingesehen werden können. Dies erschwert die Rekonstruktion der Trainingsdaten, da nicht länger die exakten Modellparameter von einzelnen Clients sichtbar sind, sondern nur aggregierte Modellparameter. Secure Aggregation erschwert so auch verschiedene Inference Attacks bzw. schützt Informationen über einzelne Clients besser, da nur die aggregierten Informationen aller Clients einsehbar sind. Differential Privacy verstärkt dazu den Schutz gegen Inference Attacks, da Trainingsdaten durch den Einsatz von Differential Privacy weniger je nach gewähltem Rauschen in den Modellparametern und Ausgaben abgebildet werden. Auch werden statistische Auswertungen über Modellausgaben, um etwas über die Trainingsdaten zu erfahren, erschwert, da diese je nach gewähltem Rauschen nicht mehr die Verteilung der Trainingsdaten widerspiegeln. Zugleich schützt Differential Privacy auch weiter vor Reconstruction Attacks, da die Modellparameter weniger die Trainingsdaten widerspiegeln.

Neben den genannten Verfahren zum Schutz der Daten beim föderalen Lernen, sollte für eine reale Umsetzung des betrachteten Anwendungsfalls zudem eine Authentifizierung als zusätzliche Schutzmaßnahme der Clients erfolgen, um zu verhindern, dass ein Angreifer während des Trainings auf Teile der Modellparameter zugreift und ungewollt am Training teilnimmt. Dies mag nicht im jeden Szenario beim föderalen Lernen sinnvoll sein, jedoch ist es hier sinnvoll, da es sich um eine feste bekannte Anzahl an Clients im Anwendungsfall handelt, denen vertraut wird. Außerdem sollte die Verbindung zwischen Server und den Clients verschlüsselt erfolgen. Sowohl für Authentifizierung als auch Verschlüsselung bietet Flower Lösungen an, die jedoch in dieser Arbeit in der Umsetzung der Einfachkeit halber nicht verwendet werden.

Die Implementierung des Anwendungsfalls wird in Python durchgeführt, da Python im Bereich des maschinellen Lernens weit verbreitet ist und viele Bibliotheken / Implementierungen für maschinelles Lernen allgemein und für föderales Lernen existieren. Zudem ist die Sprache auch grundsätzlich einfach zu nutzen.

Bei der Wahl der Bibliothek für die Umsetzung des föderalen Lernens gibt es verschiedene Möglichkeiten zur Auswahl, wovon die bekanntesten alle mit Python arbeiten. Für die Umsetzung des Anwendungsfalls in dieser Arbeit wurde das Flower Framework [12] gewählt, da es eine einfache Struktur und Anwendung hat und sich somit gut dafür eignet, um einfach die Umsetzung des Anwendungsfalls durchzuführen. Zugleich unterstützt Flower eine Vielzahl an Bibliotheken für das maschinelle Lernen wie PyTorch, TensorFlow oder scikit-learn und hat zur Nutzung von Flower selbst und mit anderen Bibliotheken eine Dokumentation. Anwendungsfälle können mit Flower als lokale Simulation durchgeführt werden, ohne getrennte Server und Clients zu benötigen. Flower bietet aber auch eine Architektur, um das föderale Lernen real über mehrere physische Geräte zu verteilen und föderales Lernen mit separaten Server und Clients durchzuführen. Außerdem bietet Flower viele Anpassungsmöglichkeiten, um Änderungen am Prozess beim föderalen Lernen vorzunehmen und ggf. weitere Implementierungen zu z.B. Schutzmechanismen einzubauen. Auch hat Flower bereits verschiedene Implementierungen zur Secure Aggregation und Differential Privacy für neuronale Netze, die für die Umsetzung des Anwendungsfalls eingesetzt werden können. Neben dieser Bibliothek gibt es noch weitere Alternativen, wie etwa NVFlare [33] oder PySyft [30], die zur Umsetzung des Anwendungsfalls genutzt werden könnten und einen größeren Funktionsumfang haben, jedoch auch komplexer sind.

Zur Umsetzung der Klassifikationsverfahren wurde für die Umsetzung der Naive Bayes Klassifikation die Implementierung der Bibliothek scikit-learn gewählt, da die Bibliothek einfach zu verwenden, gut dokumentiert und bekannt ist. Gleiches gilt auch für die Wahl der Implementierung für das LSTM-Modell. Hierzu wird die Implementierung von PyTorch verwendet, welches eines der bekanntesten Bibliotheken zur Umsetzung von neuronalen Netzen ist und zudem auch direkt kompatibel mit Flower ist.

Für die Implementierung von Secure Aggregation wird die integrierte Implementierung vom Flower Framework verwendet, welche für Verfahren, die auf festen Parametern basieren, bereits voll funktionfähig ist und als Teil von Flower auch einfach zu verwenden ist. Verwendet wird die von Flower umgesetzte Implementierung von Secure Aggregation Plus (SecAgg+) [2]. SecAgg+ bietet eine bessere Skalierbarkeit, da der Kommunikations- und Rechenaufwand bei der Secure Aggregation nur polylogarithmisch pro Client steigt und ist robuster gegenüber Ausfällen von Clients, da weniger Clients benötigt werden, um die Aggregation beim Ausfall vom Clients trotzdem erfolgreich durchführen zu können Für Differential Privacy beim LSTM-Modell wird ebenfalls die integrierte Implementierung von Flower verwendet, jedoch wird für Differential Privacy beim Naive Bayes Verfahren eine andere Implementierung benötigt, da Flower nur Differential Privacy für neuronale Netze bietet. Zur Umsetzung von Differential Privacy für das Naive Bayes Verfahren wird die IBM Differential Privacy Bibliothek [18] verwendet, welche auf der Naive Bayes Implementierung von scikit-learn aufbaut und diese Implementierung auch bereits für die Naive Bayes Klassifizierung verwendet wird. Neben der Differential Privacy Implementierung für Naive Bayes bietet die Bibliothek auch noch weitere Implementierungen für andere klassische Klassifikationsverfahren.

6 Implementierung

Die Grundlagen und das konkrete Vorhaben zusammen mit den spezifischen Anwendungsfall wurden erläutert. Im folgenden Kapitel werden nun alle Bestandteile beschrieben, welche zur Umsetzung des Anwendungsfalls benötigt werden. Darauf wird auf Besonderheiten in der Umsetzung eingegangen und weitere Details zur Umsetzung erläutert. Ziel dabei ist es, darzustellen, wie ein solcher Anwendungsfall mit föderalem Lernen implementiert werden kann. Auch wird dieses Kapitel für das darauffolgende benötigt, um zu zeigen, auf welcher Grundlage die Ergebnisse der Arbeit mit dem konkreten Anwendungsfall entstanden sind.

6.1 Datengrundlage

In diesem Abschnitt wird genauer auf die verwendeten Daten eingegangen, die zum Training der Modelle und allgemein zur Auswertung des Anwendungsfalls genutzt werden.

6.1.1 Verwendete Datensätze

Für den Anwendungsfall werden Datensätze aus E-Mail-Nachrichten benötigt. Zur Klassifizierung in Spam und Ham dieser E-Mails werden dazu sowohl Spam-Mails als auch Ham-Mails benötigt. Die E-Mails sollten in ausreichender Anzahl vorliegen, um die Modelle trainieren zu können und nach Möglichkeit realistisch auf verschiedene Clients aufteilbar sein, um zu simulieren, dass verschiedene Clients am föderalen Lernen teilnehmen, die E-Mails erhalten, die sich tendenziell auch im Inhalt unterscheiden. Die Zahl der öffentlich verfügbaren Datensätze aus Ham-E-Mails ist eher gering, da solche Nachrichten allgemein immer auch Informationen enthalten, die schützenswert sind und daher die Nachrichten nicht veröffentlicht werden können. Datensätze, die dazu verfügbar sind, enthalten dabei meist auch nur wenig Nachrichten, was das Training eines Modells zur Klassifizierung erschwert.

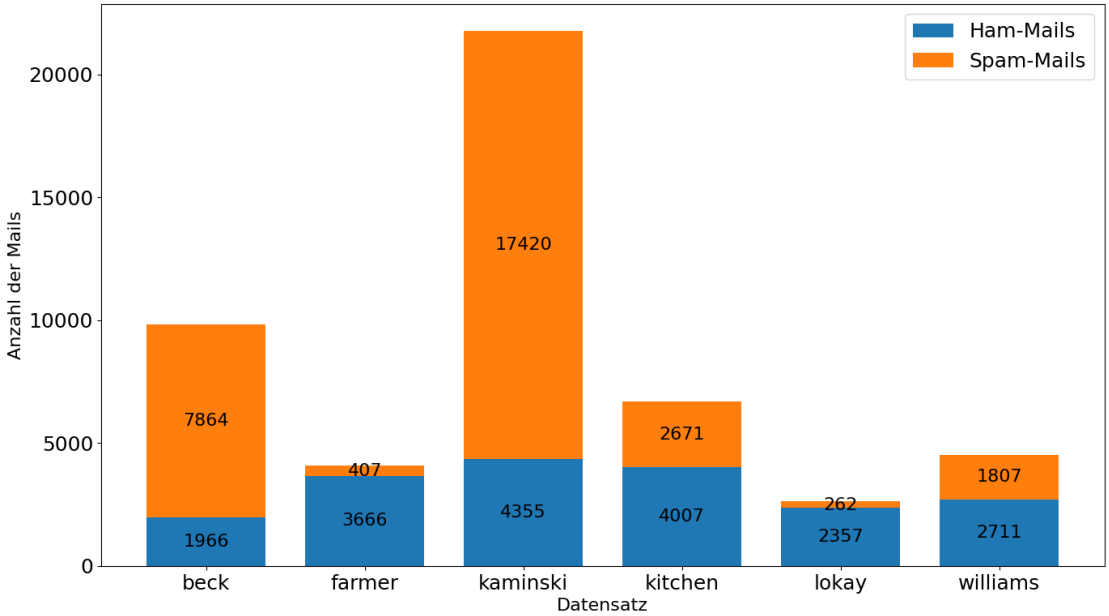

Abbildung 6.1: Überblick der Spam-Mails der einzelnen verwendeten Mail-Datensätze

Bei den verfügbaren Datensätzen in diesem Bereich gibt es einen, der besonders hervorsticht, da dieser relativ viele Nachrichten umfasst und allgemein in der Forschung in vielen Veröffentlichungen auch genutzt wird. Dabei handelt es sich um den Enron-Datensatz, der über 500.000 E-Mails in Englisch von Enron-Mitarbeitern enthält [22]. Der Datensatz wurde im Laufe von Untersuchungen der Federal Energy Regulatory Commission (FERC) der USA erstellt und durch Leslie Kaelbling aufgekauft und für Forschungszwecke veröffentlicht. Für diese Arbeit wird eine modifizierte verkleinerte Version des Enron-Datensatzes verwendet, welchem zusätzlich verschiedene Spam-Mails beigelegt sind, da dieser bereits nach Personen geordnet ist, die als einzelne Datensätze für die Clients genutzt werden können. Der verwendete Datensatz ist der Enron-Spam-Datensatz [25], welcher zum Vergleich von verschiedenen Naive Bayes-Klassifikatoren zur Spamfilterung erstellt wurde. Der Datensatz ist in sechs verschiedene Personen und deren Mail-Postfächer unterteilt und zusätzlich liegen drei verschiedene Datensätze von Spam-Mails bei. Die Spam-Mail-Datensätze wurden um Duplikate bereinigt. Der erste Spam-Datensatz (SH) besteht aus Spam-Mails des SpamAssassin Public Corpus [39] und Daten des Honeypot Projects [42]. Der zweite Spam-Datensatz (BG) besteht aus Spam-Mails, die von Bruce Guenter veröffentlicht wurden [6] und der dritte Spam-Datensatz (GP) aus E-Mails, die einer der Autoren Georgios Paliouras der Veröffentlichung zum Enron-Spam-Datensatz veröffentlicht hat. Die sechs Personen-Datensätze, welche die Ham-Mails enthalten, ha-

ben zudem Unterordner die auch als verschiedene Kategorien betrachtet werden könnten, um eine erweiterte Klassifizierung mit mehr als zwei Kategorien (Spam/Ham) durchzuführen. Dadurch würden sich je nach Person zwischen 11 bis 101 Kategorien ergeben (insgesamt 243), die jedoch nicht immer viele E-Mails zum Training umfassen. Dies wird jedoch nicht weiter in dieser Arbeit verwendet, da es die Komplexität des Anwendungsfalls weiter erhöht und dadurch diese Arbeit zu umfangreich geworden wäre.

Der Enron-Spam-Datensatz eignet sich für diesen Anwendungsfall, da er ausreichend E-Mails zur Klassifizierung erhält (über 19.000 E-Mails) und öffentlich verfügbar ist. Zugleich umfasst der Datensatz reale Ham-Mails, die auch in vielen anderen Veröffentlichungen zur Klassifizierung genutzt wurden. Die Einteilung in verschiedene Personen ermöglicht es zudem, die einzelnen Personen als Clients im föderalen Lernen zu betrachten, indem jeder Client die Ham-Mails einer Person erhält mit zusätzlichen Spam-Mails. Dadurch ist eine realistischere Verteilung der E-Mails für den Anwendungsfall möglich, wo Clients verschiedene lokale Modelle trainieren, die verschiedene Muster bei der Klassifizierung von Spam/Ham-Mails gut erkennen. Die Abbildung 6.1 stellt die Anzahl der E-Mails der sechs Personen dar, welche als einzelne Datensätze in dieser Arbeit genutzt werden, und um wie viele Spam-Mails diese jeweils ergänzt werden. Die Anzahl der Spam-Mails der einzelnen Clients ist so gewählt, dass einzelne Clients verschiedene Verteilungen von Ham/Spam haben, um beim Training zu beachten, dass manche Clients mehr oder weniger Spam-Mails erhalten als andere und dadurch eventuell auch besser lokal trainierte Modelle haben. Nach Kaspersky Lab waren etwa 45.6% aller E-Mails, die weltweit 2023 versendet wurden, Spam [21]. Ausgegangen davon, dass also im Schnitt etwa 40-50% der E-Mails Spam sind, wird der Anteil der Spam-Mails je nach Clients variiert. Dazu werden die sechs Clients anhand der Anzahl der Ham-Mails aufgeteilt:

- Ein Client (Beck) mit niedriger Anzahl an Ham-Mails und hohem Anteil an Spam-Mails (80% der E-Mails sind Spam)
- Ein Client (Lokay) mit niedriger Anzahl an Ham-Mails und niedriger Anzahl an Spam-Mails (10% der E-Mails sind Spam)
- Ein Client (Williams) mit mittlerer Anzahl an Ham-Mails und mittlerer Anzahl an Spam-Mails (40% der E-Mails sind Spam)
- Ein Client (Farmer) mit hoher Anzahl an Ham-Mails und niedriger Anzahl an Spam-Mails (10% der E-Mails sind Spam)

- Ein Client (Kitchen) mit hoher Anzahl an Ham-Mails und mittlerer Anzahl an Spam-Mails (40% der E-Mails sind Spam)
- Ein Client (Kaminski) mit hoher Anzahl an Ham-Mails und hoher Anzahl an Spam-Mails (80% der E-Mails sind Spam)

Natürlich wären auch andere Variationen möglich, jedoch ermöglicht diese Verteilung sowohl einen Mangel an Spam-Mails als auch einen Überschuss an Spam-Mails zu betrachten. Die Clients erhalten zudem ihre Spam-Mails von festgelegten Datensätzen der drei Spam-Datensätze (als Alternative zur komplett zufälligen Auswahl der Spam-Mails aus allen drei Spam-Datensätzen für die Clients), damit die Clients unterschiedliche Arten von Spam-Mails erhalten (ausgegangen wird davon, dass die Spam-Mails zwischen den drei Spam-Datensätzen sich auch allgemein unterscheiden und nicht ähnliche Spam-Mails repräsentieren). Die Tabelle 6.1 stellt dazu dar, aus welchem der drei Spam-Datensätze die Clients Spam-Mails beziehen.

Tabelle 6.1: Überblick der einzelnen Mail-Datensätze

Person	Spam-Mails aus GP	Spam-Mails aus BG	Spam-Mails aus SH
Kaminski	12914	4506	0
Beck	0	5353	2511
Farmer	0	0	407
Kitchen	0	0	2671
Lokay	0	0	262
Williams	0	0	1807
Insgesamt	12914	9859	7658

6.1.2 Vorbereitung der Daten

Um die E-Mails zum Training und zum Testen der Klassifizierung verwenden zu können, werden diese zunächst noch weiter vorverarbeitet, um eine Verarbeitung der E-Mails zu ermöglichen und bessere Ergebnisse bei der Klassifizierung zu erzielen. Dazu werden die E-Mails, welche als einzelne Dateien vorliegen, zunächst eingelesen und geparst, wozu die Python-Bibliothek mail-parser[1] genutzt wird, um vor allem die Betreffzeile und den Inhalt der E-Mails leichter zu extrahieren. Dazu werden auch E-Mails aussortiert, die weniger als 10 Wörter enthalten, da bei diesen E-Mails meist zu wenig Informationen vorhanden sind,

[1] https://github.com/SpamScope/mail-parser

um überhaupt eine sinnvolle Klassifikation anhand des Inhalts zu ermöglichen. Zudem werden dadurch auch E-Mails aussortiert, die z.B. nur aus Bildern bestehen und keinen Textinhalt enthalten. Aus den Inhalt der E-Mails werden zudem HTML-Tags entfernt, da sie keine inhaltliche Bedeutung haben und um den reinen Text ohne Formatierung zu erhalten, falls der Inhalt mit HTML formatiert wurde, wofür die Python-Bibliothek BeautifulSoup[2] genutzt wird. Sollte der Inhalt einer Mail mit HTML formatiert worden sein, könnten zudem Sonderzeichen als HTML-Sonderzeichen enthalten sein, welche mit der Standardbibliothek html von Python in Unicode Zeichen umgewandelt werden. Danach wird eine Tokenisierung des Mailinhalts aus Betreff und dem eigentlichen Inhalt durchgeführt, um Tokens für die weitere Verarbeitung zu erhalten. Dazu werden auch alle Wörter mit weniger als 3 Zeichen entfernt, um irrelevante Informationen zu entfernen. Auch werden Satzzeichen und alle Wörter entfernt, die in einer Liste von englischen Stopwörtern enthalten sind und als irrelevante Informationen betrachtet werden. Zudem werden Wörter entfernt, die Zahlen oder Satzzeichen enthalten, um z.B. Domains im E-Mail-Inhalt zu ignorieren und das Vokabular nicht unnötig zu vergrößern. Um das Vokabular weiter zu vereinfachen, wird eine Lemmatisierung der übrig gebliebenen Wörter durchgeführt. Dadurch werden einzelne Wörter auf ihre Grundform zusammengefasst[3]. Dies wird dem Stemming gegenüber bevorzugt, da die Lemmatisierung auch den Kontext und die Bedeutung der Wörter erhält, was für die Klassifizierung sinnvoll ist, da eine genauere Unterscheidung der Mails anhand des Kontextes und der Bedeutung der Wörter möglich ist. Schließlich werden alle Wörter in Kleinschreibung umgewandelt, um das Vokabular weiter zu verringern. Mithilfe eines trainierten Word2Vec-Modells werden anschließend Vektoren aus den einzelnen Tokens gebildet. Aus diesen Vektoren wird für das Naive Bayes-Verfahren ein Durchschnittsvektor gebildet, um eine gemeinsame Eingabe zu haben, die den Inhalt einer Mail repräsentiert. Ein LSTM-Modell kriegt wiederum die einzelnen Vektoren als Eingabe. Für die Verarbeitung der E-Mails durch ein LSTM-Modell wird zudem die Anzahl der Tokens auf 500 begrenzt, da in der verwendeten Python-Bibliothek PyTorch alle Eingabesequenzen gleich lang sein müssen und die längsten E-Mails im Datensatz über 40.000 Wörter umfassen, wodurch alle Eingabesequenzen so lang sein müssten. Zu kurze Nachrichten werden dazu mit einem temporären Padding aufgefüllt, welches jedoch von PyTorch ignoriert wird. Nachrichten, die länger als 500 Tokens sind, werden auf 500 Tokens begrenzt. Die Zahl von 500 Tokens wurde gewählt, da sie zum einen genug Wörter umfasst, um eine vernünftige Klassifikation zu ermöglichen und zum anderen, wie in der Abbildung 6.2 zu sehen ist, der Großteil der E-Mails

[2] https://www.crummy.com/software/BeautifulSoup/
[3] https://de.wikipedia.org/wiki/Lemma_(Lexikographie)#Lemmatisierung

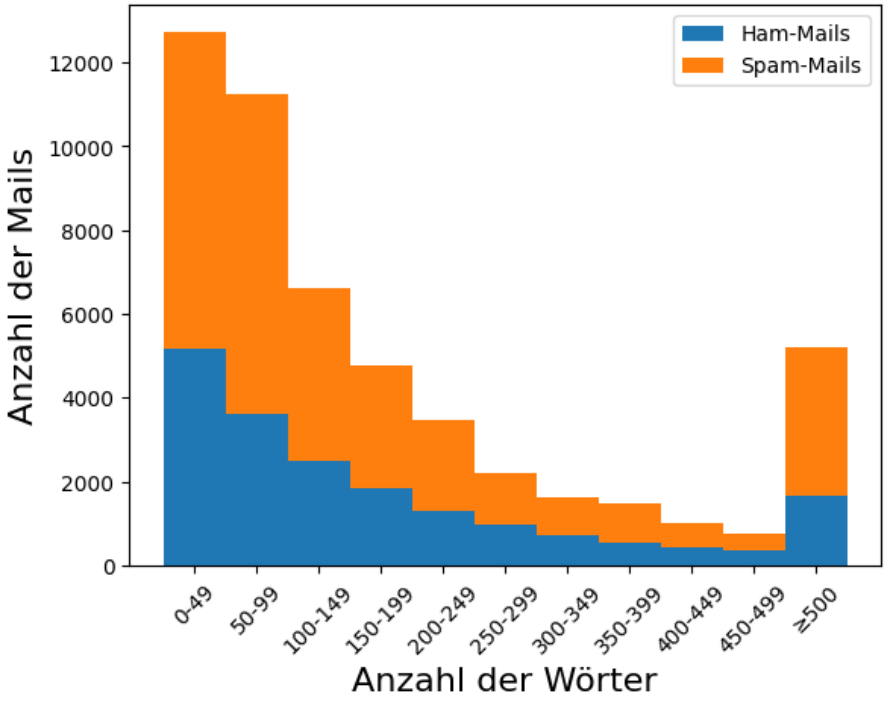

Abbildung 6.2: Verteilung der Mail-Längen in Enron-Spam

weniger als 500 Wörter umfasst (andere und eventuell auch niedrigere Begrenzungen, die diese Eigenschaften erfüllen, wären hier auch möglich).

6.1.3 Word2Vec-Modell

Für die Arbeit wurde ein Word2Vec-Modell aus allen verwendeten E-Mails des Enron-Spam-Datensatzes mit vorverarbeiteten Texten, wie zuvor beschrieben, trainiert. Alternativ wurde auch betrachtet, ein vortrainiertes Word Embedding zu verwenden. Eine Möglichkeit dazu sind die vortrainierten Wikipedia2Vec Word Embeddings, welche auf Texten, die auf Wikipedia verfügbar sind, trainiert wurden [43]. Diese Word Embeddings sind jedoch deutlich größer als selbstrainierte Word Embeddings, da sie auch ein deutlich größeres Vokabular abbilden und die Vektorgröße größer ist, wodurch die Trainingszeit und der Arbeitsspeicherbedarf deutlich ansteigt. Daher wurde für die Umsetzung das selbstrainierte Modell genommen. In einer praktischen Umsetzung würde das Vorgehen, wie in dieser Arbeit jedoch zu zwei Problemen führen. Zum einen müsste ein solches Modell häufiger aktualisiert werden, um mögliche neue Wörter, die zuvor nicht in E-

Mails vorkamen, zu beachten. Alternativ können unbekannte Wörter auch nicht beachtet werden, was jedoch die Genauigkeit der Klassifikation beeinträchtigen könnte. Außerdem wurde das Word2Vec-Modell hier aus allen vorhandenen Daten trainiert. Praktisch würden die Daten jedoch verteilt bei den Clients liegen und sollten nicht zentral gesammelt werden, weshalb auch das Word2Vec-Modell föderal trainiert werden müsste, um die Daten zu schützen. Dies reicht jedoch nicht unbedingt aus, da auch das trainierte Word2Vec-Modell, die Trainingsdaten z.B. über ein Membership Inference Angriff preisgeben könnte [38], weshalb hierzu weitere Maßnahmen erforderlich wären, die jedoch über den Umfang der Arbeit hinaus gehen und hier nicht weiter betrachtet werden. Das Risiko ist bei einem Word2Vec-Modell allerdings geringer als bei komplexeren Modellen wie neuronale Netze, da die Vektoren von einzelnen Wörtern, welche die einzigen Parameter des Modells sind, auch unabhängig der Häufigkeit des Wortes in den Trainingsdaten von allen anderen Wörtern in den Trainingsdaten beeinflusst werden. Eine simplere und datenschutzfreundlichere Lösung wäre daher trotzdem, ein unabhängig vom Anwendungsfall vortrainiertes Modell oder andere Alternativen zur Textrepräsentation zu verwenden.

6.2 Architektur

In diesem Kapitel wird kurz ein Einblick in die verwendete Architektur zur Umsetzung des Anwendungsfalls und in Form eines Klassendiagramms in die Implementierung gegeben. Die Abbildung 6.3 stellt dar, wie die einzelnen Clients und der Server untereinander kommunizieren und in welcher Umgebung diese bei der Auswertung zum Anwendungsfall in dieser Arbeit verteilt werden. Zentral ist dabei im Flower Framework der SuperLink-Server, welcher als Koordinator beim föderalen Lernen dient. Dieser wird hier direkt über die Kommandozeile per RPC gesteuert, um neue Trainingsdurchläufe zu starten und die Projekte dazu zu verteilen. Die SuperNodes bei den Clients registrieren sich dazu beim SuperLink und erhalten die Anfragen zu den Trainingsdurchläufen und die Projekte. In dieser Umsetzung wird die Umgebung des Servers und der Clients in einem separaten Prozess gestartet (ServerApp und ClientApp). Alternativ können die Projekte auch direkt beim SuperLink und den SuperNodes in einen Unterprozess gestartet werden. Die benötigten Abhängigkeiten der Projekte sind dazu bereits in den Umgebungen vorinstalliert.

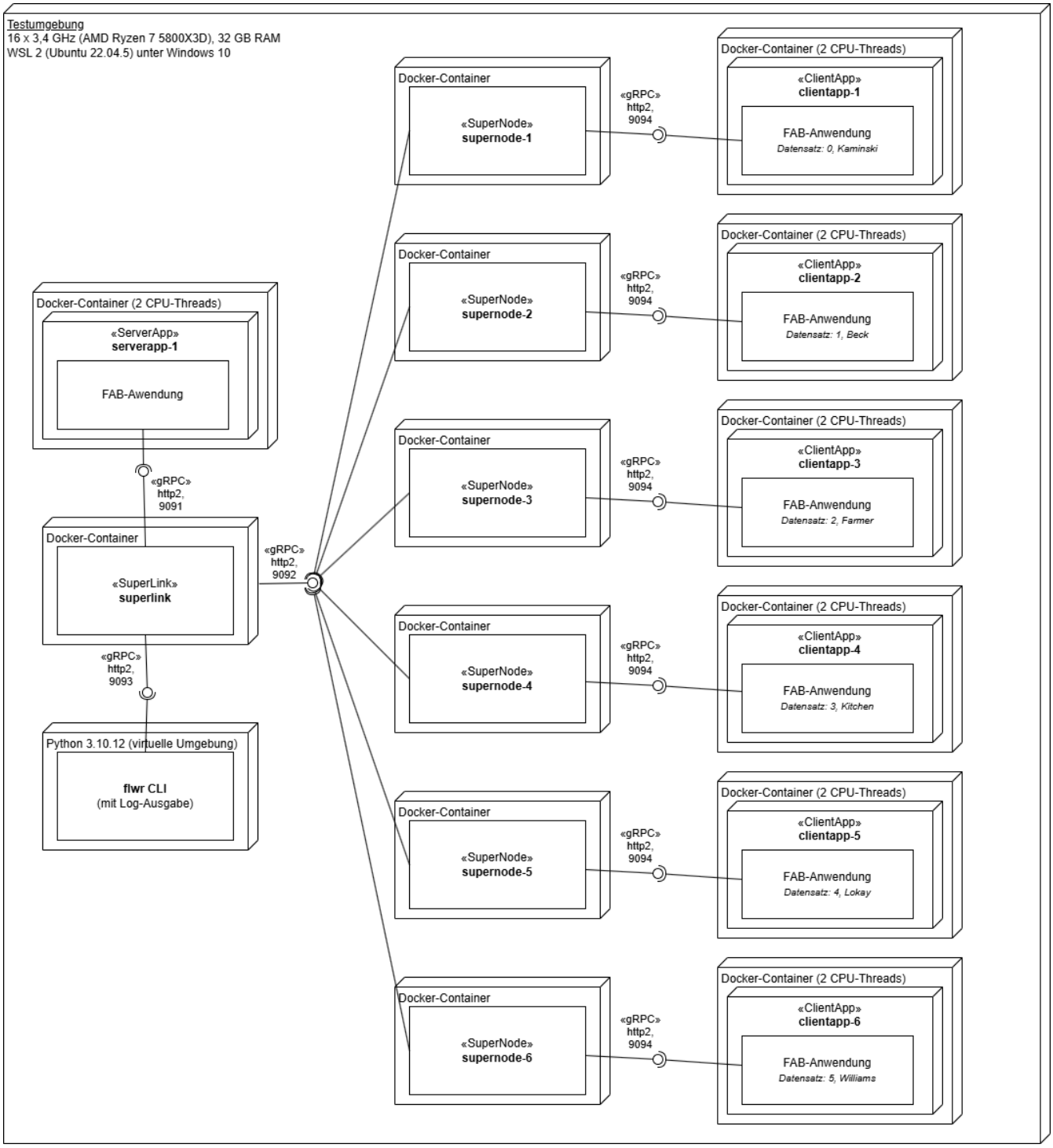

Abbildung 6.3: Verteilungssicht der Umgebung zur Auswertung beim föderalen Lernen

Die Abbildung 6.4 stellt zudem die Implementierung als Klassendiagramm dar. Die einzelnen Python-Module werden dabei auch als Klassen mit dem Stereotyp «Module» dargestellt. Die Implementierung ist allgemein in ein jeweiliges Projekt für das verwendete Klassifikationsverfahren aufgeteilt. Zudem gibt es ein zentrales `flclassificationdata`-Projekt, welches zentrale Methoden rund um das Laden und die Verarbeitung der Daten enthält sowie Methoden zum Laden der Metriken und zur Auswertung dieser. Das Projekt für das Random Forest Verfahren ist hierbei nicht enthalten, da die Umsetzung gescheitert ist und es im weiteren Verlauf der Umsetzung danach größere Umstrukturierungen gab. Es folgt aber grundsätzlich dem gleichen Schema der anderen Klassifikationsverfahren. Die jeweiligen Projekte zu den Klassifikationsverfahren selbst sind primär nach den Vorgaben vom Flower Framework strukturiert. Ein Client lädt dabei jeweils seinen Datensatz und eventuell auch der Server, um eine zentrale Evaluation der Modelle durchzuführen, was in dieser Arbeit durchgeführt wurde, um die Modelle auf allen Daten zu evaluieren, jedoch in einer praktischen Umsetzung in der Art nicht möglich ist bzw. anders umgesetzt werden müsste, da dann kein zentraler Zugriff auf alle Daten existiert. Die Clients umfassen jeweils drei zentrale Methoden. Die Methode `fit` nimmt Modellparameter entgegen und trainiert das Modell dazu mit dem lokalen Datensatz des Clients und gibt die Modellparameter zurück. Die Methode `evaluate` nimmt wiederum die Modellparameter entgegen, die aggregiert wurden, und evaluiert bei jedem Client, wie effektiv das Modell beim jeweiligen Client ist. Der Server definiert wiederum, mit welcher Methode die Modellparameter und Evaluationsmetriken aggregiert werden sollen und definiert ggf. mit `get_eval_fn` eine Methode, um die Modellparameter zentral zu evaluieren.

6.3 Umgebung der Clients und des Servers

Die einzelnen Clients und der Server laufen für die Durchführung der Szenarien, wie zuvor in der Abbildung 6.3 beschrieben, isoliert in Docker-Containern. Dies ermöglicht eine einfachere Ermittlung der Metriken, Skalierung der Client-Anzahl und Beschränkung der für einen Client verfügbaren Ressourcen. Jeder Client und auch der Server erhält dabei 2 CPU-Threads zur Berechnung und 2 GB Arbeitsspeicher, welche über Docker beschränkt werden. Eine komplette Isolation der CPU-Threads für die jeweiligen Clients/dem Server findet nicht statt, jedoch laufen während der Durchführung der Szenarien keine weiteren Prozesse neben den benötigten Systemprozessen auf dem Host-System, wodurch der Einfluss anderer Prozesse auf die Durchführung minimiert werden soll.

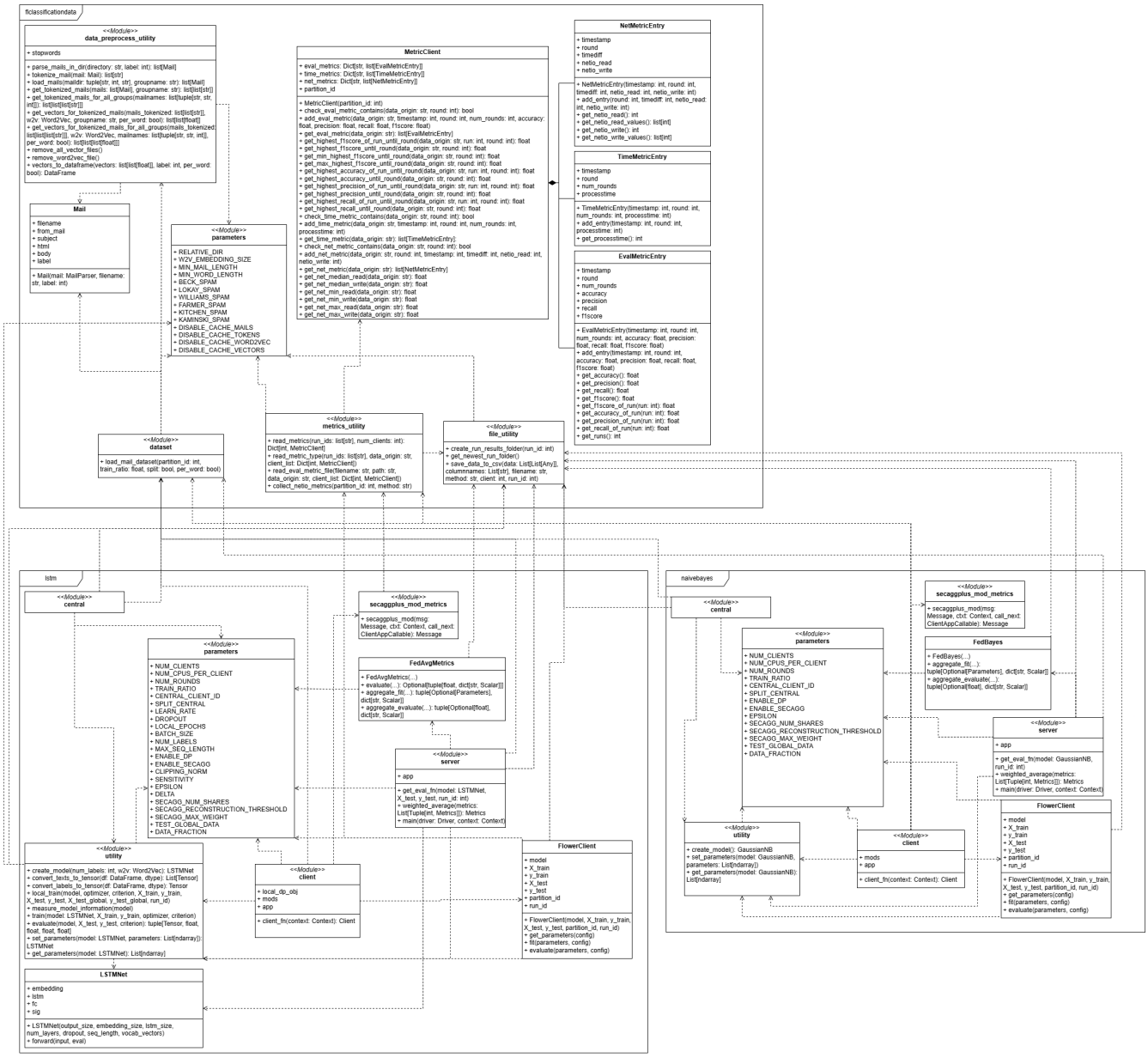

Abbildung 6.4: Klassendiagramm zur Implementierung des Anwendungsfalls

6.4 Weitere Anmerkungen zur Umsetzung

Für die Umsetzung des föderalen Lernens mit einem Naive Bayes-Modell wurde eine eigene Implementierung zur Aggregation der Parameter geschrieben (hier als FedBayes bezeichnet), die allgemein dem Prinzip von FedAvg folgt, die Modellparameter zu sammeln und einen gewichteten Durchschnitt zu bilden. Ähnliches lässt sich auch für andere Verfahren umsetzen, vorausgesetzt diese haben klare Modellparameter, die aggregiert werden können. Die Implementierung von föderalem Lernen mit Random Forest ist gescheitert, da kein Verfahren umgesetzt werden konnte, um die einzelnen Random Forests der Clients zu einem gemeinsamen Random Forest zu aggregieren. Ausprobiert wurde dabei, zufällige Entscheidungsbäume aus jeden Random Forest jedes Clients auszuwählen und mit gleichen Anteil in einen gemeinsamen Random Forest zu übernehmen. Zudem konnte auch keine passende Implementierung zur Differential Privacy für Random Forest gefunden werden, die auch korrekt funktioniert, weshalb insgesamt aus zeitlichen Gründen in der Umsetzung die Auswertung mit Random Forest nicht weiter betrachtet wird.

In der Abbildung 6.5 wird das implementierte LSTM-Modell dargestellt. Beliebig andere Modelle mit anderen Parametern sind zum Zweck der Klassifikation für den Anwendungsfall natürlich möglich (etwa die Variation der Anzahl der LSTM-Blöcke, wovon hier zwei verwendet werden), jedoch hat sich dieses LSTM-Modell als geeignet erwiesen, eine ausreichende Effektivität zu erreichen und dabei nicht zu rechenaufwändig zu sein. Bei dieser Umsetzung ist wichtig zu beachten, dass für das Embedding Layer die Wortvektoren von Word2Vec verwendet werden und daher das Embedding Layer selbst nicht trainiert wird. Bei der Verwendung der Differential Privacy wird daher auch sichergestellt, dass das Embedding Layer durch das Rauschen nicht verändert wird, da hier auch keine Modellparameter enthalten sind, die durch das Training möglicherweise schützenswerte Informationen enthalten könnten.

Um die benötigten Metriken, die im nächsten Kapitel beschrieben werden, für die Szenarien zu sammeln, ist zudem die Implementierung an einigen Stellen weiter angepasst worden. Um die Rechenzeiten bei der Aggregation zu erfassen, wurde die Implementierung zu FedAvg und FedBayes um die Erfassung der Rechenzeit erweitert.

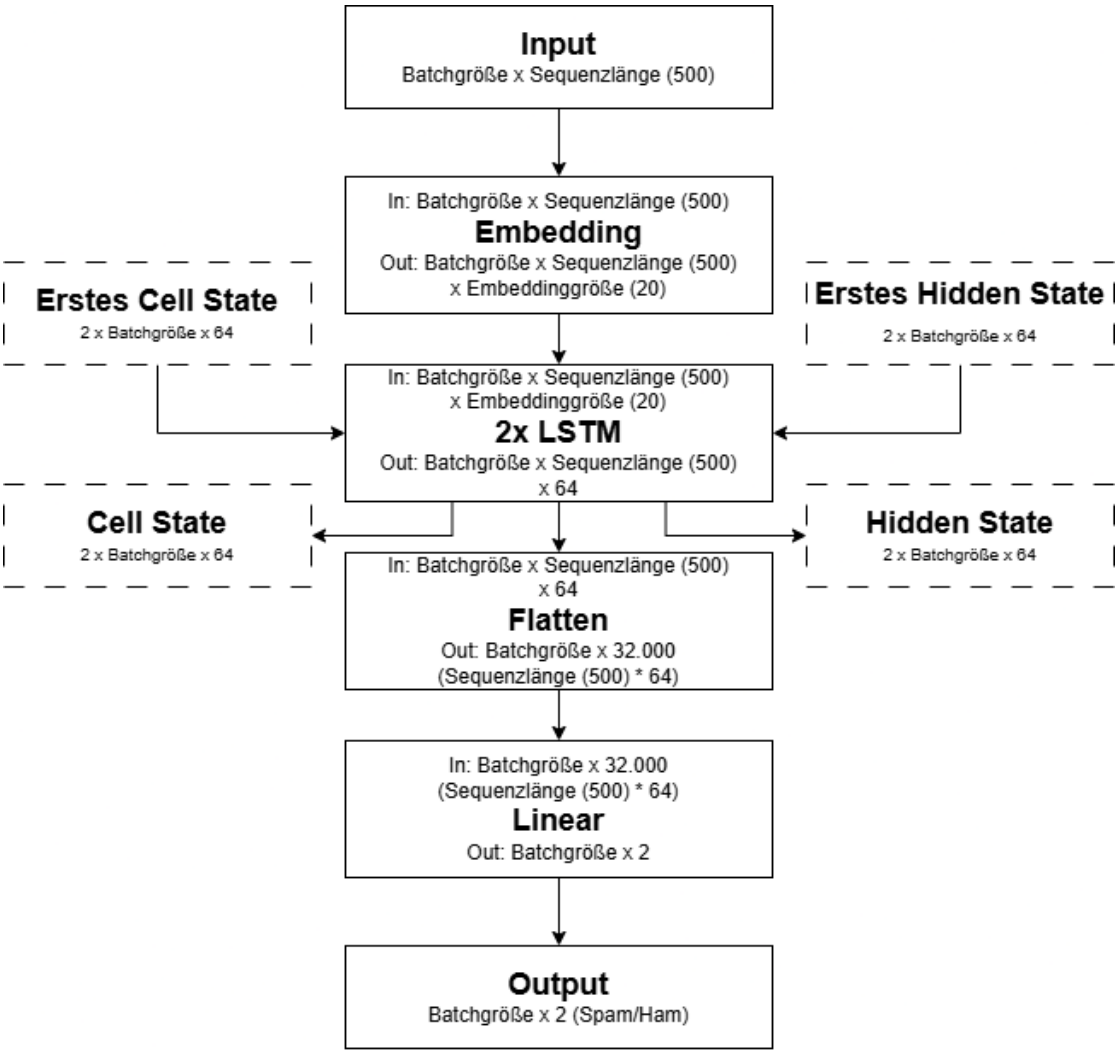

Abbildung 6.5: LSTM-Modell für den Anwendungsfall

7 Messungen und Ergebnisse

Im vorherigen Kapitel wurden die Einzelheiten zur konkreten Implementierung des Anwendungsfalls erläutert. Im folgenden Kapitel sollen nun zunächst die Ergebnisse der Arbeit in den verschiedenen Szenarien vorgestellt werden. Dafür wird zuerst das Vorgehen bei den Messungen erläutert, also welche Metriken aufgenommen und wie diese bestimmt werden. Danach werden die Messergebnisse für die einzelnen Szenarien präsentiert.

7.1 Vorgehen bei den Messungen

Zum Evaluieren des föderalen Lernens in den verschiedenen Szenarien sind verschiedene Metriken notwendig, um den Erfolg des Lernens zu bewerten. Um die Effektivität der Klassifikation dabei zu bestimmen, können die Metriken Accuracy, Precision, Recall und F1-Score betrachtet werden [17]. Bei der Klassifikation wird die Korrektheit der Klassifikation der Elemente allgemein in die vier Kategorien True Positive (TP, positiv klassifizierte Elemente, die real positiv sind), True Negative (TN, negativ klassifizierte Elemente, die real negativ sind), False Positive (FP, positiv klassifizierte Elemente, die nicht real positiv sind) und False Negative (FN, negativ klassifizierte Elemente, die real nicht negativ sind) eingeteilt. Die Accuracy gibt dazu bei einer binären Klassifikation an, wie viele Elemente der klassifizierten Elemente korrekt (entsprechen auch wirklich der Klasse) klassifiziert wurden (auf den Anwendungsfall bezogen also: Welcher Anteil der E-Mails wurde korrekt als Spam oder Ham klassifiziert). Bestimmt wird die Accuracy durch:

$$Accuracy = \frac{Korrekt\ klassifizierte\ Elemente}{Alle\ klassifizierten\ Elemente} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

Die Precision gibt an, wie viele der Elemente, die als positiv klassifiziert wurden, auch wirklich positiv sind (daher welcher Anteil der als Spam klassifizierten E-Mails ist auch

wirklich Spam). Bestimmt wird die Precision durch:

$$Precision = \frac{Klassifizierte\ real\ positive\ Elemente}{Alle\ positiv\ klassifizierten\ Elemente} = \frac{TP}{TP + FP} \tag{7.2}$$

Der Recall gibt an, welcher Anteil der tatsächlich positiven Elemente auch als positiv klassifiziert wurden (welcher Anteil der Spam-Mails wurde als Spam erkannt). Bestimmt wird der Recall durch:

$$Recall = \frac{Klassifizierte\ real\ positive\ Elemente}{Alle\ positiven\ Elemente} = \frac{TP}{TP + FN} \tag{7.3}$$

Im Anwendungsfall ist von den drei Metriken vor allem die Precision wichtig, da vermieden werden soll, dass Ham-Mails als Spam klassifiziert werden und lieber Spam-Mails nicht erkannt werden sollten, als das Ham-Mails als Spam klassifiziert werden. Zugleich sollte der Recall-Wert aber auch erhöht werden, um den Anteil des erkannten Spams zu erhöhen. Accuracy kann dazu als Metrik betrachtet werden, um sowohl Precision als auch Recall zu berücksichtigen. Jedoch kann die Accuracy nicht gut mit einer ungleichmäßigen Klassenverteilung umgehen, da z.B. bei wenig Spam in einem Datensatz und der Klassifizierung, dass alle E-Mails kein Spam sind, die Accuracy hoch sein würde, da nur wenige Elemente falsch klassifiziert wurden. Der F1-Score ist dazu eine Metrik, die sowohl Precision als auch Recall gleichermaßen gewichtet und dadurch ein besseres Maß für die Klassifikation in dem Anwendungsfall dieser Arbeit ist. Bestimmt wird der F1-Score durch:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP + TN}{2TP + FP + FN} \tag{7.4}$$

Die genannten Metriken zur Bestimmung der Effektivität der Klassifikation werden bei jedem Client gemessen. Zudem wird beim Server der aggregierte Durchschnitt der Metriken über alle Clients bestimmt und eine Messung zentral mit allen Testdaten durchgeführt. Diese Testdaten enthalten alle vorhandenen Ham- & Spam-Mails, wodurch die Evaluation der Modelle darüber ein noch klareres Bild über die Effektivität der Klassifikation gibt, welches auch einen Teil an Spam-Mails enthält, der in keinen der Datensätze der Clients enthalten ist. Auch beim zentralen Lernen wird die Effektivität über die Daten aus Sicht eines Clients und mit allen Daten bestimmt, wenn das jeweilige Szenario dort unterscheidet.

Um die Effizienz der Klassifikation zu bestimmen, wird die benötigte Rechenzeit beim föderalen Lernen und zentralen Lernen gemessen. Zur Messung der Rechenzeit wird die

Methode `process_time_ns` aus der Python-Standardbibliothek `time`[1] verwendet, da so nur die benötigte Rechenzeit des Prozesses der CPU gemessen wird und nicht weitere Zeit, die abgesehen vom eigentlichen Prozess benötigt wird, beachtet wird. Hierdurch wird der Einfluss anderer Prozess und Verzögerungen durch Schreib- und Lesevorgänge auf die gemessene Rechenzeit minimiert. Zugleich ist bei der Messung der Effizienz zu beachten, das eine exakte Bestimmung der Rechenzeit nicht möglich ist, da andere Prozesse auf dem System die Rechenzeit beeinflussen können. Dies ist jedoch auch nicht nötig, da die Rechenzeit hier viel mehr als Vergleichswert innerhalb dieser Arbeit genutzt wird und sich zumindest so eine Tendenz zur Verbesserung oder Verschlechterung der Effizienz durch Änderungen in den Szenarien erkennen lässt. Zur Messung der Rechenzeit wird die benötigte Zeit für das Training der Modelle und für die Evaluation separat gemessen. Zudem wird beim föderalen Lernen noch die benötigte Zeit für die Aggregation beim Server gemessen, welche dort Teil des Trainings ist. Mögliche Latenzen zwischen den Clients und Server werden bei der Messung nicht beachtet, da Clients und Server im Testszenario sich alle auf einem Rechner befinden und so keine nennenswerten Latenzen entstehen. Neben der Rechenzeit wird zudem noch die gesamte Laufzeit beim Training betrachtet, um ein grobes Maß für die benötigten Zeiten zu geben, die in der Rechenzeit nicht beachtet werden konnten wie die benötigte Zeit zur Kommunikation.

Das genutzte Datenvolumen der Clients wird beim föderalen Lernen ebenfalls erfasst. Da die Clients alle in Docker Containern ausgeführt werden, wird die Differenz des aufgezeichneten Netzwerkverkehrs des Containers zwischen zwei Runden als Datenvolumen pro Runde betrachtet. Abgesehen vom eigentlichen Datenvolumen durchs Training führt der SuperLink-Server beim Flower Framework noch jede Sekunde Ping-Anfragen an die Clients durch, deren Datenvolumen aber nicht abgezogen werden bei den Messungen, da dieses zum einen zu gering ist und zum anderen nicht ganz konsistent (Etwa 196 Bytes empfangen und 77 Bytes versendet pro Sekunde). Zudem beginnt die Aufzeichnung des Datenvolumens erst, nachdem die Clients die Anwendung erhalten haben, wodurch das Datenvolumen für die Kommunikation in diesem Teil vor Beginn der ersten Runde fehlt.

Zudem wird die maximal genutzte Menge an Arbeitsspeicher beim Training der Modelle und bei der Evaluation gemessen, indem die mit den Python tracemalloc Module der maximal genutzte Arbeitsspeicher bei jedem Client während des Trainings und der Evaluation gemessen wird. Die Größe der Datensätze und der Modelle werden in den

[1] https://docs.python.org/3/library/time.html#time.process_time

Szenarien auch erfasst, um einen Vergleichswert für die genutzten Daten mit den Datenvolumen und der Arbeitsspeichernutzung zu haben.

7.2 Szenarien

7.2.1 Szenario 1: Ausgangsszenario

Im Ausgangsszenario sind zunächst Differential Privacy und Secure Aggregation deaktiviert. Alle sechs Clients nehmen dabei am föderalen Lernen teil. Die Daten sind wie zuvor beschrieben zwischen den einzelnen Clients aufgeteilt und bestehen aus der in Abschnitt 6.1.1 angegebenen Menge Ham/Spam-Mails. Um eine geeignete Rundenanzahl für die weitere Untersuchung zu finden, wird beim föderalen Lernen zunächst mit 50 Runden trainiert. Bei der Verwendung von Naive Bayes ist die Rundenanzahl irrelevant, da ein Modell hier nach einer Runde bereits trainiert ist. Allgemein werden zudem alle Ergebnisse in diesem Szenario, wie auch in allen folgenden, über mehrere Durchläufe gemittelt, um mögliche Schwankungen beim Training auszugleichen. Auch wird in allen Szenarien das gleiche Word2Vec-Modell genutzt, damit die Ergebnisse der einzelnen Szenarien besser vergleichbar sind. Praktisch können in den einzelnen Szenarien eine effektivere oder uneffektivere Klassifikation durch ein anderes Word2Vec-Modell erreicht werden und auch durch eine andere Textvorverarbeitung. Das zentrale Lernen wird in diesem Szenario auch untersucht, wobei die Daten aller Clients zusammen genutzt werden, um ein Modell zu trainieren. Trainiert wird hier ebenfalls über 50 Runden.

Die Abbildung 7.1 zeigt die Effektivität der Klassifikation beim föderalen Lernen und zentralen Lernen im Ausgangsszenario. Die Werte wurden über fünf Runden gemittelt und die vertikale Linie in der Abbildung markiert die Runde, zu der nach der Evaluation beim Server über alle Daten der höchste F1-Score erreicht wurde. Zudem zeigt die Tabelle 7.1 die maximal erreichten Werte der Effektivität bei der LSTM-Klassifikation. Am Verlauf des Trainings ist erkennbar, dass zentrales Lernen hier bereits von Runde 1 deutlich besser startet, da dem zentralen Lernen in diesem Szenario alle Daten zur Verfügung stehen und so direkt ein besseres Modell trainiert werden kann. Im Verlauf des Trainings nähert sich das föderal trainierte Modell jedoch immer mehr der Effektivität des zentral trainierten Modells an und erreicht schließlich ein ähnliche Effektivität wie das zentral trainierte Modell. Für die weiteren Szenarien in der Auswertung wird eine geringere Rundenanzahl gewählt, um die Auswertung schneller durchführen zu können.

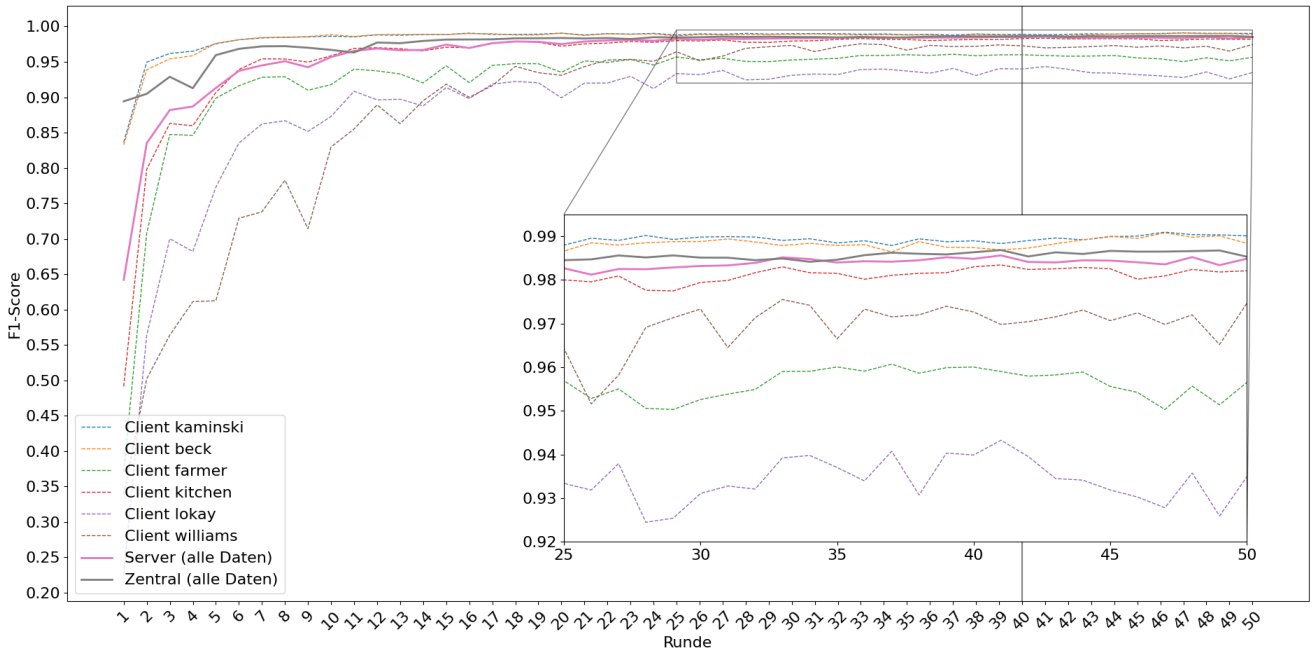

Abbildung 7.1: Effektivität (F1-Score) der LSTM-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen)

Dazu wird 30 Runden als Rundenanzahl gewählt, da sich hier bereits das zentral und das föderal trainierte Modell in der Effektivität fast gleichen.

Tabelle 7.1: Szenario 1: Messwerte zur Effektivität der LSTM-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen)

Client	Accuracy	Precision	Recall	F1-Score
kaminski	99,18%	99,18%	99,18%	99,18%
beck	99,25%	99,25%	99,25%	99,25%
farmer	96,25%	97,26%	96,25%	96,50%
kitchen	98,67%	98,68%	98,67%	98,67%
lokay	94,66%	96,46%	94,66%	95,13%
williams	97,77%	97,80%	97,77%	97,77%
Server (aggregierte Ergebnisse)	98,19%	98,67%	98,19%	98,25%
Server (alle Daten)	98,67%	98,67%	98,67%	98,67%
Zentral (alle Daten)	98,81%	98,81%	98,81%	98,81%

Bei der Effektivität der Klassifikation mit Naive Bayes sieht es etwas anderes aus. Zunächst wird das Naive Bayes-Modell nur über eine Runde beim föderalen Lernen trainiert, da ein Naive Bayes-Modell auch nach einer weiteren Trainingsrunde mit den selben Trai-

ningsdaten die selben Parameter hat. Dies liegt daran, dass die Modellparameter beim Naive Bayes nicht an die Effektivität als z.B. Loss-Wert angepasst werden. Die Tabelle 7.2 zeigt die im Durchschnitt über fünf Durchläufe beste erreichte Effektivität der Klassifikation mit Naive Bayes beim föderalen Lernen und zentralen Lernen. Im Vergleich zum LSTM-Modell kommt das Naive Bayes-Modell jedoch beim föderalen Lernen mit einem F1-Score von 85,36% über alle Daten nicht an die Effektivität des zentralen Lernens über alle Daten mit einem F1-Score von 87,65% heran. Dies hängt wahrscheinlich auch von der Aggregationsmethode ab, wobei hier nur in eigener Implementierung wie zuvor beschrieben, ähnlich zu FedAvg die Modellparameter der Clients gewichtet gemittelt werden. Auffällig ist auch, dass die Effektivität der einzelnen Clients deutlich mehr schwankt als beim LSTM-Modell. Der Client mit dem williams Datensatz sticht dort besonders hervor mit einem F1-Score von nur 60,02%.

Tabelle 7.2: Szenario 1: Messwerte zur Effektivität der Naive Bayes-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen)

Client	Accuracy	Precision	Recall	F1-Score
kaminski	88,45%	91,60%	88,45%	89,19%
beck	86,52%	90,85%	86,52%	87,50%
farmer	85,77%	92,50%	85,77%	87,96%
kitchen	89,15%	89,14%	89,15%	89,09%
lokay	83,97%	90,70%	83,97%	86,28%
williams	60,84%	67,73%	60,84%	60,02%
Server (aggregierte Ergebnisse)	85,18%	88,97%	85,18%	85,92%
Server (alle Daten)	85,27%	85,60%	85,27%	85,36%
Zentral (alle Daten)	87,51%	88,55%	87,51%	87,65%

Die Abbildung 7.2 stellt zudem die Effizienz beim Training des LSTM-Modells anhand der benötigten Rechenzeit dar. Fürs föderale Lernen wurde in jeder Runde die längste benötigte Rechenzeit aller Clients zusammen mit der benötigten Rechenzeit zur Aggregation genommen. Sowohl beim föderalen als auch beim zentralen Lernen bleibt die benötigte Rechenzeit pro Runde dabei relativ konstant. Über fünf Durchläufe gemittelt zeigt sich dabei, dass zentrales Lernen mit 6262 Sekunden benötigter Rechenzeit für 50 Runden 52,7% mehr Rechenzeit benötigt als föderales Lernen mit etwa 4100 Sekunden. Dieser Unterschied zeigt sich auch bei der insgesamt benötigten Laufzeit, wo föderales Lernen für 50 Runden 3174 Sekunden benötigt hat und zentrales Lernen 3444 Sekunden. Zugleich ist der Unterschied hier allerdings deutlich kleiner, da die benötigte Zeit zur Kommunikation zwischen Client und Server nicht in der Rechenzeit enthalten ist beim föderalen Lernen. Anzumerken ist noch, dass die gesamte benötigte Rechenzeit beim fö-

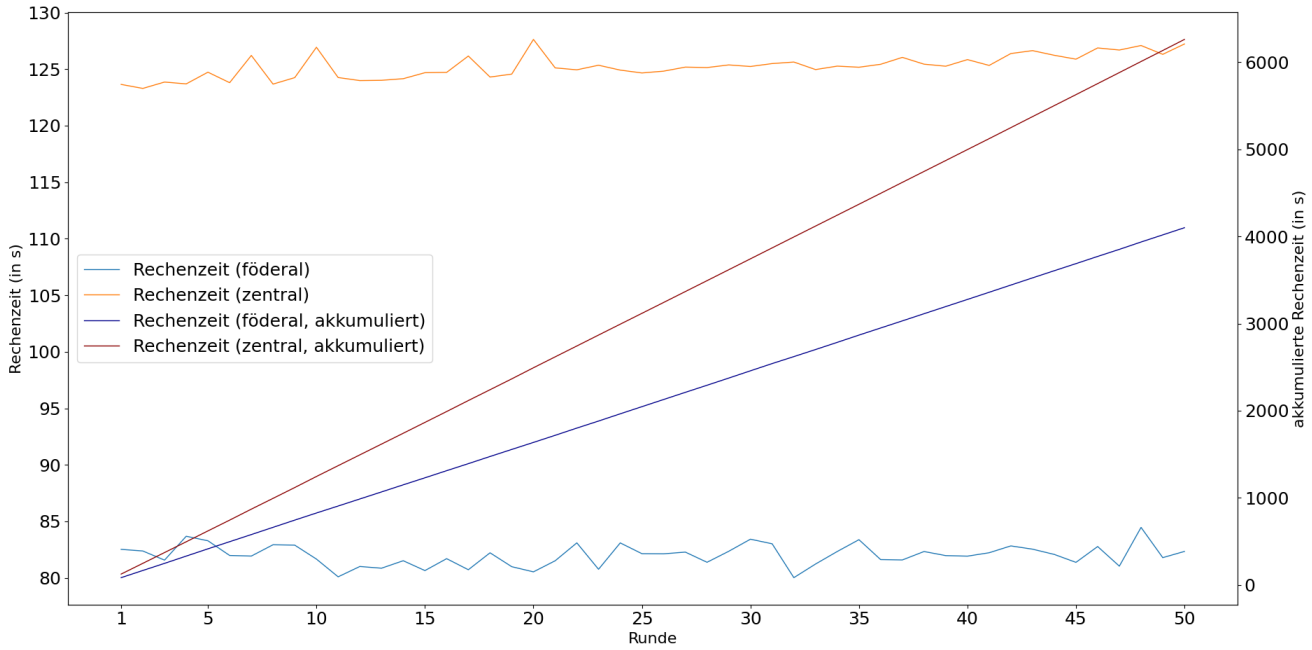

Abbildung 7.2: Effizienz (Rechenzeit) der LSTM-Klassifikation beim föderalen Lernen und zentralen Lernen (Durchschnitt von fünf Durchläufen)

deralen Lernen, wenn alle Clients zusammen gezählt werden, bei 12062 Sekunden liegt und damit deutlich höher ist als die Rechenzeit beim zentralen Lernen. Dieser Unterschied ist im Hinblick auf die Energieeffizienz des Trainings noch interessant, was jedoch in dieser Arbeit nicht weiter betrachtet wird. Die benötigte Rechenzeit bei der Evaluation der Modelle ist beim föderalen Lernen in jeder Runde auch deutlich kürzer als beim zentralen Lernen, welches 101% mehr Rechenzeit benötigt, da auch hier beim föderalen Lernen pro Client ein deutlich kleinerer Teil der Daten evaluiert wird als in diesem Szenario beim zentralen Lernen, wo alle Daten genutzt werden (Föderales Lernen: 270 Sekunden, Zentrales Lernen: 543 Sekunden).

Ein Effizienzvergleich zwischen dem föderalen und zentralen Lernen ist beim Training des Naive Bayes-Modells nicht sinnvoll möglich, da das Training des Naive Bayes-Modells sehr wenig Rechenzeit pro Runde benötigt und zudem nur eine Runde trainiert wird. Da die Messungen jedoch einen groben Rahmen zur Einordnung bieten, werden sie trotzdem kurz erwähnt. Föderales Lernen hat hierbei für eine Runde im Durchschnitt von fünf Durchläufen 18,4 Millisekunden benötigt und zentrales Lernen 7,6 Millisekunden. Die Evaluation beim föderalen Lernen hat etwa 69,8 Millisekunden an Rechenzeit benö-

tigt und beim zentralen Lernen 16,4 Millisekunden. Bei der gemessenen Gesamtlaufzeit zeigt sich dieser Unterschied hingegen nicht mehr und geht wieder zur Tendenz vom LSTM-Modell mit 6,02 Sekunden für das föderale Lernen und deutlich längeren 8,99 Sekunden für das zentrale Lernen. Im Allgemeinen beläuft sich der Unterschied zwischen dem föderalen und zentralen Lernen beim Naive Bayes-Modell nur auf den vorhandenen Kommunikations- und den trivialen Aggregationsaufwand beim föderalen Lernen und der Ladezeit für die Daten, was vermutlich primär zur längeren Zeit beim zentralen Lernen führt.

Die Tabelle 7.3 stellt zudem die Größe der Datensätze in MB nach der Vorverarbeitung dar. Hier wird unterschieden zwischen der Größe, wenn der Datensatz aus den Indizes für einzelne Vektoren besteht, was für die LSTM-Modelle verwendet wird, und der Größe, wenn der Datensatz aus Durchschnittsvektoren besteht, die aus allen Wörtern einer Mail gebildet haben und als Eingabedaten für die Naive Bayes-Klassifikation genutzt werden. Das trainierte Word2Vec-Modell dazu ist etwa 9,17 MB groß.

Tabelle 7.3: Szenario 1: Größe der Datensätze in MB

Datensatz	Größe (Einzelvektoren)	Größe (Durchschnittsvektor)
kaminski	22,26 MB	2,26 MB
beck	11,38 MB	1,02 MB
farmer	2,95 MB	0,42 MB
kitchen	13,91 MB	0,69 MB
lokay	3,03 MB	0,27 MB
williams	4,02 MB	0,47 MB
Insgesamt	57,55 MB	5,15 MB

Je nach Datensatz und Verfahren unterscheidet sich zudem die Arbeitsspeichernutzung beim Training. Die Tabelle 7.4 stellt die maximal erreichte Arbeitsspeichernutzung beim LSTM-Modell und beim Naive Bayes-Modell dazu dar und gibt so einen Einblick zur benötigten Arbeitsspeichermenge für den Anwendungsfall. Daneben wurde auch das Datenvolumen beim föderalen Lernen pro Runde erfasst. Aus den Messungen ist zu erkennen, dass das Datenvolumen, für alle Clients grundsätzlich unabhängig der Runde konstant bleibt. Der Median des empfangenen und gesendeten Datenvolumens bei jedem Client pro Runde beträgt 9,13 MB empfangen (minimal 9,12 MB und maximal 10,21 MB) und 4,54 MB gesendet (minimal 4,53 MB und maximal 4,55 MB) beim LSTM-Modell. Die Modellparameter des LSTM-Modells selbst sind dazu 4,52 MB groß, was etwa der gesendeten Größe auch entspricht. Beim Naive Bayes-Modell ist das gemessene Datenvolumen

deutlich geringer mit einem Median des empfangenen und gesendeten Datenvolumens bei jedem Client pro Runde bei 56,75 KB empfangen (minimal 55,89 KB und maximal 57,55 KB) und 6,66 KB gesendet (minimal 5,48 KB und maximal 7,71 KB). Die Modellparameter des Naive Bayes-Modells sind dazu 2,06 KB groß.

Tabelle 7.4: Szenario 1: Maximale Arbeitsspeichernutzung

Datensatz	RAM (LSTM)	RAM (NB)
kaminski	539,38 MB	415,34 MB
beck	499,64 MB	384,57 MB
farmer	504,49 MB	388,92 MB
kitchen	608,98 MB	467,27 MB
lokay	509,15 MB	391,77 MB
williams	493,11 MB	379,97 MB
Zentral (alle Daten)	710,87 MB	568,70 MB

7.2.2 Szenario 2: Föderales Lernen mit Secure Aggregation

In diesem Szenario wird föderales Lernen mit Secure Aggregation verwendet, um die Aggregation der Modellparameter zu schützen. Bei Betrachtung der maximal erreichten Effektivität von fünf Durchläufen mit 30 Runden, ausgewertet über alle Daten, fällt auf, dass föderales Lernen mit Secure Aggregation einen F1-Score von 98.37% beim LSTM-Modell erreicht (mit kleinsten Wert von 98.24% und größten Wert von 98.57% unter den Durchläufen), jedoch ohne Secure Aggregation einen F1-Score von 98.49% erreicht (mit kleinsten Wert von 98.35% und größten Wert von 98.60% unter den Durchläufen). Der Unterschied ist dabei relativ gering, sodass dieser eventuell auf Schwankungen beim Training zurückzuführen sind. Eine andere Möglichkeit ist, dass die Secure Aggregation-Implementierung zu geringfügigen Veränderungen an den Modellparametern führt, indem etwa die Parameter gerundet werden. Bei gleicher Auswertung der Effektivität von fünf Durchläufen mit einem Naive Bayes-Modell fällt auf, dass hier die Effektivität identisch zwischen dem Training mit Secure Aggregation und ohne bei einem F1-Score von 85,36% ist. Dies bestätigt, dass die Secure Aggregation an sich keine Auswirkung auf die Effektivität hatte und bei der LSTM-Umsetzung nur geringfügige Veränderungen durch die Implementierung entstehen.

Die maximale Arbeitsspeichernutzung steigt durch den Einsatz von Secure Aggregation hierbei nicht an. Das gemessene Datenvolumen verdoppelt sich in etwa im Vergleich

zum föderalen Lernen ohne Secure Aggregation beim LSTM-Modell. Der Median des empfangenen und gesendeten Datenvolumens bei jedem Client pro Runde beträgt mit Secure Aggregation 18,32 MB empfangen (minimal 13,79 MB und maximal 19,77 MB) und 9,11 MB gesendet (minimal 9,10 MB und maximal 10,50 MB) beim LSTM-Modell. Der Anstieg des Datenvolumens mit Secure Aggregation ist beim Naive Bayes-Modell hingegen deutlich größer. Beim Naive Bayes-Modell beträgt der Median des empfangenen und gesendeten Datenvolumens bei jedem Client pro Runde mit Secure Aggregation 178,21 KB empfangen (minimal 175,43 KB und maximal 179,23 KB) und 52,51 KB gesendet (minimal 49,80 KB und maximal 54,14 KB).

Die benötigte Rechenzeit beim Training der Modelle ändert sich nicht durch Secure Aggregation, da die Modelle immer noch gleich trainiert werden. Jedoch führt Secure Aggregation zu einer längeren Laufzeit, da dies zusätzliche Kommunikationsrunden zwischen Client und Server erfordert. Dadurch liegt die Laufzeit im Durchschnitt von fünf Durchläufen bei 30 Runden beim LSTM-Modell mit Secure Aggregation bei etwa 2454 Sekunden und ohne bei etwa 1903 Sekunden, was einer Steigerung von etwa 29% entspricht. Beim Naive Bayes-Modell im Durchschnitt von fünf Durchläufen bei einer Runde liegt die Laufzeit mit Secure Aggregation bei etwa 24,69 Sekunden und ohne bei 6,02 Sekunden, was einen Anstieg um 310% durch Secure Aggregation entspricht. Da beim Naive Bayes-Modell wie schon zuvor das Training selbst kaum Zeit benötigt, zeigt sich hier der Anstieg durch die Secure Aggregation besonders stark. Die benötigte Rechenzeit bei der Evaluation der Modelle wird durch Secure Aggregation auch nicht beeinflusst, da Secure Aggregation nur bei der Aggregation der Modellparameter beim Training genutzt wird und keine direkten Auswirkungen auf die trainierten Modelle hat.

7.2.3 Szenario 3: Föderales Lernen mit Differential Privacy

Für die Umsetzung von Differential Privacy mit LSTM-Modellen wird die Local Differential Privacy Implementierung des Flower Frameworks verwendet, welche (ϵ, δ)-Differential Privacy garantiert und den Gaußschen Mechanismus für das Rauschen auf die Modellparameter anwendet. Der δ-Wert wird bereits im Voraus auf 10^{-5} festgelegt, da auch jede einzelne Mail im Anwendungsfall schützenswert ist und daher ein δ-Wert gewählt wird, der sehr gering ist, um das Risiko so gering wie möglich zu halten, dass ϵ-Differential Privacy nicht eingehalten wird. Bei der Umsetzung des Anwendungsfalls mit föderalem Lernen sind weitere unerwartete Herausforderungen bei der Verwendung von Differential Privacy zum Schutz der Daten aufgetreten. Bei der Umsetzung der Klassifikation

mit einem LSTM-Modell ist ein Problem bei der Wahl von kleinen Werten für ϵ aufgetreten. Das LSTM-Modell ist grundsätzlich sehr tief, da 500 Wörter mit Vektoren der Größe 20 für jede Mail verarbeitet wurden, wodurch das LSTM-Module 500x durchlaufen wird. Im Anwendungsfall hat dies nicht zu erkennbaren Problemen geführt, jedoch verursacht das hinzugefügte Rauschen durch Differential Privacy hingegen Probleme. Je nach gewähltem Epsilon wird dadurch das Rauschen so stark, dass es den normalen Wertebereich der Modellgewichte überschreitet. Die Gewichte werden dadurch so groß, dass wiederum die daraus resultierenden Gradienten beim Gradientenabstieg durch die 500 Durchläufe so groß werden, dass die Gradienten über den Wertebereich der Datentypen hinaus gehen und NaN (nicht als Zahl darstellbar) als Wert annehmen. Dadurch werden auch die Gewichte auf NaN aktualisiert und der Trainingsdurchlauf scheitert. Maßgeblich wird dies auch durch die L2-Sensitivität des Gausschen Rauschens, welches in der Flower-Implementierung verwendet wird, gesteuert. Die L2-Sensitivität wird je nach Anwendungsfall festgelegt und da das Rauschen hierbei direkt auf den Modellgewichten angewandt wird, wird die L2-Sensitivität an den Wertebereich der Modellparameter angepasst. Normalerweise wird die L2-Sensitivität dann so gewählt, dass sie die höchstmögliche Werteveränderung in einem Durchlauf abbildet. Durch das zuvor beschriebene Gradientenproblem reicht dies jedoch nicht aus, da bei etwa $\epsilon = 0,1$ die Gewichte zu groß werden. Als mögliche Lösung wurde überprüft, welche Möglichkeiten es gibt, die Werte im Modell möglichst klein zu halten. Eine Normalisierung der Eingabevektoren wurde ausprobiert, was jedoch zu keiner Verbesserung geführt hat. Zudem wurde ein Clipping der Gradienten beim Training versucht, damit die Gradienten nicht zu groß werden. Der Ansatz funktioniert teilweise, geht jedoch bei zu großer L2-Sensitivität nicht, da die Gradienten innerhalb einer Runde so stark ansteigen, dass die resultierenden Gradienten bereits NaN annehmen, selbst wenn in vorherigen Runden geclippt wurde, wodurch Clipping der Werte keine Auswirkung hat. Naheliegend ist noch, die eigentliche Sequenzlänge der Eingabe zu verringern, damit weniger als 500 Wörter verarbeitet werden müssen. Dies wurde jedoch nicht weiter untersucht, da es die Vergleichbarkeit mit den anderen Szenarien beeinträchtig, weil dort ebenfalls die Sequenzlänge angepasst werden müsste. Stattdessen wurde die L2-Sensitivität auf einen ausreichenden Wert von 0,005 festgelegt, um eine weitere Untersuchung des Szenarios mit Differential Privacy zu ermöglichen.

Die Abbildung 7.3 zeigt die Auswertung der Effektivität anhand des F1-Scores beim föderalen Lernen mit Differential Privacy für unterschiedliche Epsilon-Werte beim LSTM-Modell. Gezeigt wird der Durchschnitt der höchsten F1-Scores der Durchläufe. Hieran ist zu erkennen, dass, wie zu erwarten, bei steigendem Rauschen durch niedrigere Epsilon-

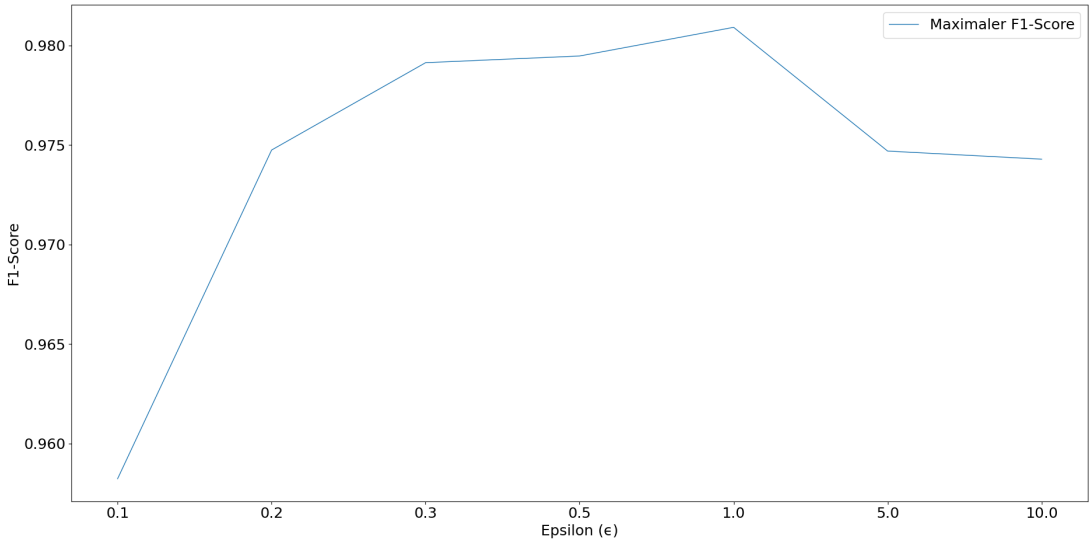

Abbildung 7.3: Vergleich der Effektivität bei unterschiedlichen Epsilon-Werten beim LSTM-Modell (Durchschnitt von fünf Durchläufen)

Werte die Effektivität abnimmt, wobei von den verglichenen Intervallen bis $\epsilon = 0,3$ der Effektivitätsverlust relativ gering ist (F1-Score von 97.91% bei $\epsilon = 0,3$ im Vergleich zu 98.09% bei $\epsilon = 1,0$). Unerwartet ist jedoch, dass die Effektivität über $\epsilon = 1,0$ auch abnimmt. Dies könnte damit zusammenhängen, dass der verwendete Gaußsche Mechanismus nur für $\epsilon <= 1$ garantiert, dass (ϵ, δ)-Differential Privacy eingehalten wird [10, S.261]. Ein weiterer möglicher Grund könnte sein, dass aufgrund von nur fünf durchgeführten Durchläufen pro Epsilon-Wert zufällig niedrigere Messungen bei $\epsilon = 5,0$ und $\epsilon = 10,0$ dabei waren als bei $\epsilon = 1,0$, was auch dadurch bekräftigt wird, dass der Großteil der Messungen ähnlich zu den Messungen bei $\epsilon = 1,0$ ist. Eine genauere Erklärung dazu wurde jedoch nicht gefunden. Die Abbildung 7.4 zeigt eine weitere Sicht auf föderales Lernen mit Differential Privacy, indem die Effektivität über mehrere Runden mit und ohne Differential Privacy bei unterschiedlichen Epsilon-Werten verglichen wird. Zentral ist hier erkennbar, dass die Effektivität der Klassifikation mit Differential Privacy größtenteils unter der Effektivität ohne Differential Privacy liegt.

Die Abbildung 7.5 zeigt zudem die Auswertung der Effektivität anhand des F1-Scores beim föderalen Lernen mit Differential Privacy für unterschiedliche Epsilon-Werte auch beim Naive Bayes-Modell. Das Naive Bayes-Modell wurde hierzu, anders als beim ersten Szenario, über 30 Runden trainiert, da durch das Rauschen durch Differential Privacy die Modellparameter sich bei gleichen Eingabedaten trotzdem verändern. Beim Naive Bayes-

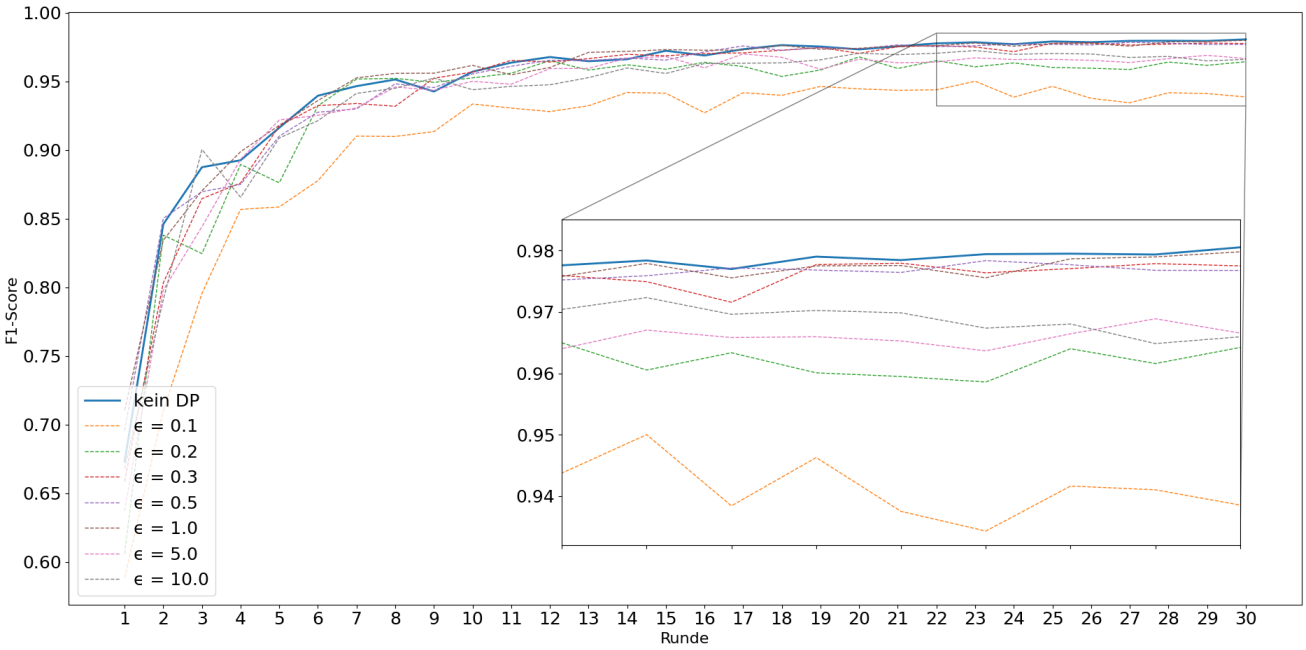

Abbildung 7.4: Verlauf der Effektivität bei unterschiedlichen Epsilon-Werten beim LSTM-Modell (Durchschnitt von fünf Durchläufen)

Modell ist erkennbar, dass das hinzugefügte Rauschen teilweise sogar positive Effekte auf die Effektivität hat, wenngleich auch das Naive Bayes-Modell nicht an die Effektivität des LSTM-Modells heran kommt. Von $\epsilon = 1,0$ bis etwa $\epsilon = 0,5$ steigt die Effektivität an. Darunter nimmt die Effektivität wieder ab, da hier Modellparameter zu sehr beeinträchtigt werden. Über $\epsilon = 1,0$ steigt zudem die Effektivität auch wieder an, was im Gegensatz zu der Umsetzung beim LSTM-Modell auch ein Bereich ist, der theoretisch verwendet werden kann, wenn auch auf Kosten der Privatsphäre der Daten, da der Laplace-Mechanismus für das Rauschen verwendet wird. Eine Erklärung für die steigende Effektivität zwischen $\epsilon = 0,5$ und $\epsilon = 1,0$ wurde nicht gefunden. Eine naheliegende Erklärung dazu ist jedoch, dass das hinzugefügte Rauschen eine mögliche Überanpassung des Modells auf die Trainingsdaten beim Training reduziert. Das Modell kann so besser generalisieren, was zu einer Verbesserung der Effektivität führt. Die Abbildung 7.6 zeigt dazu für $\epsilon = 0,3$ als gewählten Wert bei dem allgemein der Effektivitätsverlust noch gering ist, aber die Trainingsdaten möglichst geschützt werden, dass die Effektivität durch das zufällige Rauschen nach einigen Trainingsrunden in den meisten Fällen die Effektivität des Modells ohne Differential Privacy übertrifft. Die Effektivität des Modells ohne Differential Privacy aus dem ersten Szenario wird dazu als horizontale Linie dargestellt

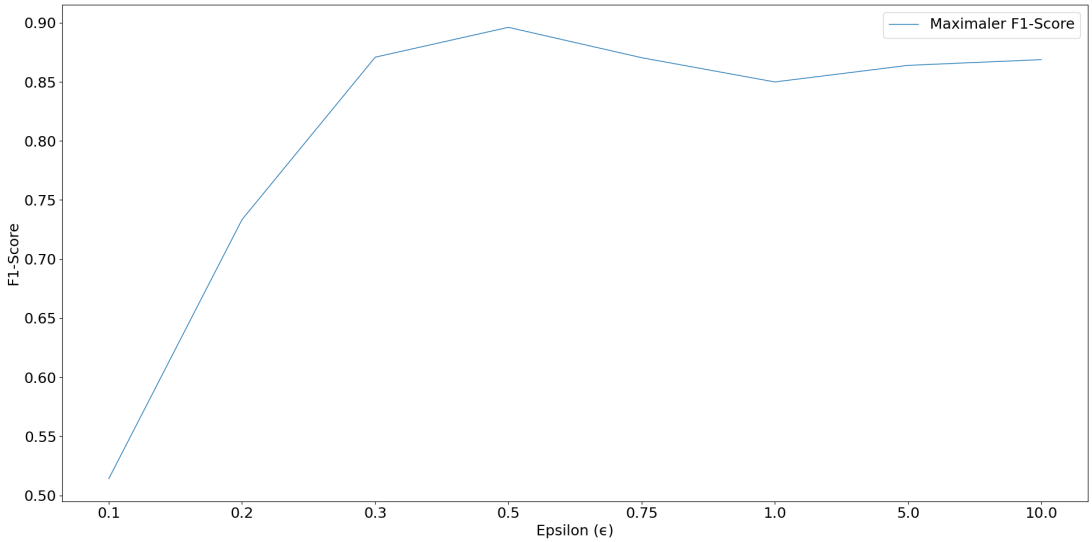

Abbildung 7.5: Vergleich der Effektivität bei unterschiedlichen Epsilon-Werten beim Naive Bayes-Modell (Durchschnitt von fünf Durchläufen)

und die Verläufe der einzelnen Durchläufe enden jeweils an dem Punkt, wo der höchste F1-Score erreicht wurde.

Die Verwendung von Differential Privacy hat auch wie Secure Aggregation keine direkte Auswirkung auf die Evaluation der Modelle, da Differential Privacy nur die Modellparameter beim Training verändert.

7.2.4 Szenario 4: Föderales Lernen mit Differential Privacy & Secure Aggregation

In diesem Szenario wird kurz die Kombination von Differential Privacy und Secure Aggregation betrachtet. Differential Privacy wird mit $\epsilon = 0,3$ und den weiteren in Szenario 3 festgelegten Parametern verwendet. Die Effektivität der Klassifikation ändert sich dabei nicht im Vergleich zum Szenario 3 mit Differential Privacy, da die Secure Aggregation keine direkte Auswirkung darauf hat (abgesehen von der zuvor erwähnten geringfügigen Veränderung beim LSTM-Modell). Zudem verändert sich Rechenzeit beim Training wie auch nur mit Differential Privacy nicht bedeutend, da Differential Privacy im Verhältnis zum eigentlichen Training keine bedeutende Auswirkung auf die Rechenzeit hat. Die Gesamtlaufzeit entspricht daher auch in etwa der Laufzeit mit nur Secure Aggregation bei

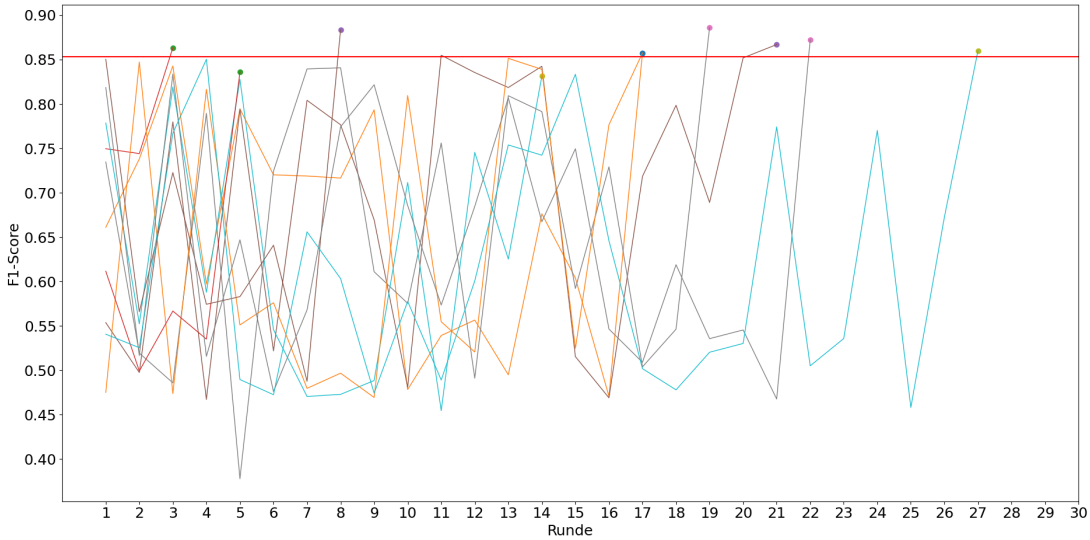

Abbildung 7.6: Verlauf der Effektivität von zehn Durchläufen bei Epsilon 0,3 beim Naive Bayes-Modell

2424 Sekunden für 30 Runden beim LSTM-Modell mit Secure Aggregation. Ein Vergleich mit dem Naive Bayes-Modell dazu wurde an dieser Stelle nicht weiter betrachtet.

7.2.5 Szenario 5: Zentrales Lernen mit geringer Datenmenge

In diesem Szenario wird nur das zentrale Lernen betrachtet und mit dem föderalen Lernen verglichen. Dabei werden beim zentralen Lernen anders als zuvor nur die Daten für einen Client genutzt. Dieses Szenario wird betrachtet, um, anders als beim ersten Szenario, wo beim zentralen Lernen mit allen Daten trainiert wurde, zu untersuchen, wie effektiv zentrales Lernen im Vergleich zum föderalen Lernen ist, wenn der Datenschutz auch strikt eingehalten wird und jedem Client wirklich nur seine Daten zur Verfügung stehen. Die Abbildung 7.7 zeigt die Entwicklung der Effektivität über 30 Runden im Durchschnitt von fünf Durchläufen dazu dreimal für zentrales Lernen mit jeweils nur einem Datensatz, für das föderale Lernen mit Secure Aggregation und Differential Privacy, mit $\epsilon = 0,3$ und den zuvor festgelegten Parametern, und für nur das föderale Lernen aus dem ersten Szenario. Die Effektivität wird bei jedem Client über alle Daten ausgewertet. Wie zu erwarten, entwickelt sich die Effektivität des zentralen Lernen mit jeweils nur einem Datensatz unter der Effektivität des föderalen Lernens, wobei die Clients mit den größten Datensätzen tendenziell die höchste Effektivität erreichen. Am meisten ist dies beim

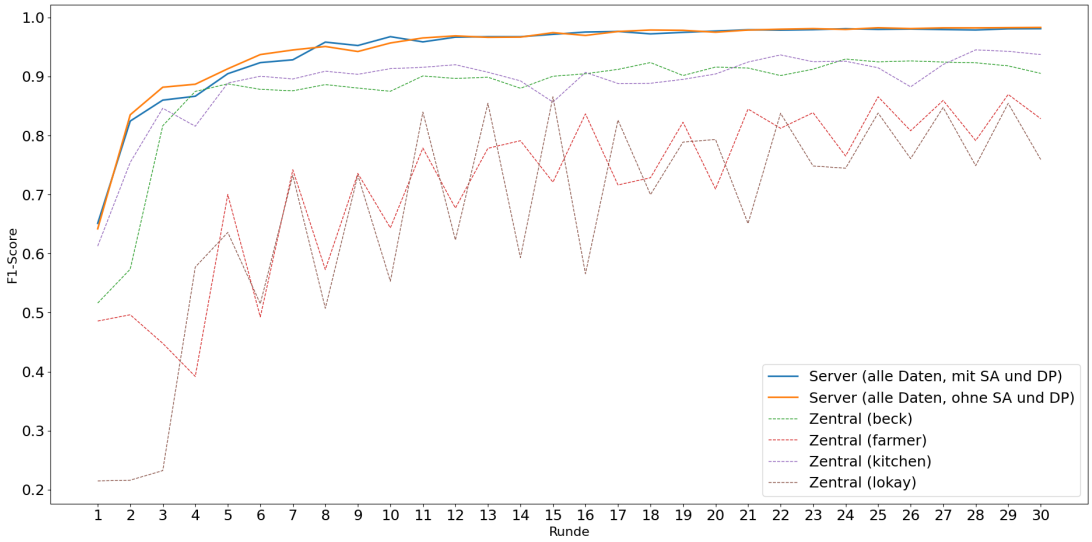

Abbildung 7.7: Effektivität (F1-Score) der LSTM-Klassifikation beim zentralen Lernen mit geringer Datenmenge im Vergleich zum föderalen Lernen (Durchschnitt von fünf Durchläufen)

Client mit dem lokay-Datensatz sichtbar, der die niedrigste Effektivität erreicht und die größten Schwankungen beim Training hat. Zugleich erreichen aber alle Clients mit allen Datensätzen einen F1-Score von über 80%, da selbst der kleinste Datensatz etwa 2500 E-Mails umfasst und allgemein die zugrundeliegenden Ham- und Spam-Datensätze insgesamt zu homogen sind bzw. nur eine begrenzte Gruppe an E-Mails abbilden, sodass auch aus einem kleinen Datensatz bereits viel gelernt wird.

Anders sieht es hingegen beim Naive Bayes-Modell aus, wozu in Abbildung 7.8 die durchschnittlich erreichte maximale Effektivität über fünf Durchläufe dargestellt ist. Eine Tendenz zu höherer Effektivität bei einem größeren Datensatz ist auch hier erkennbar, jedoch liegt die Effektivität des föderalen Lernens nicht über den Einzelergebnissen jedes Datensatzes evaluiert auf allen Daten. Auszugehen ist davon, dass dies am Vorgehen bei der Aggregation der lokalen Modelle beim föderalen Lernen liegt und hier somit noch ein besseres Vorgehen genutzt werden müsste.

7.2.6 Szenario 6: Verringerte Anzahl an Clients

In diesem Szenario wird abgesehen von der Verringerung der Clientanzahl beim föderalen Lernen auch Secure Aggregation angewandt und Differential Privacy mit $\epsilon = 0,3$ genutzt

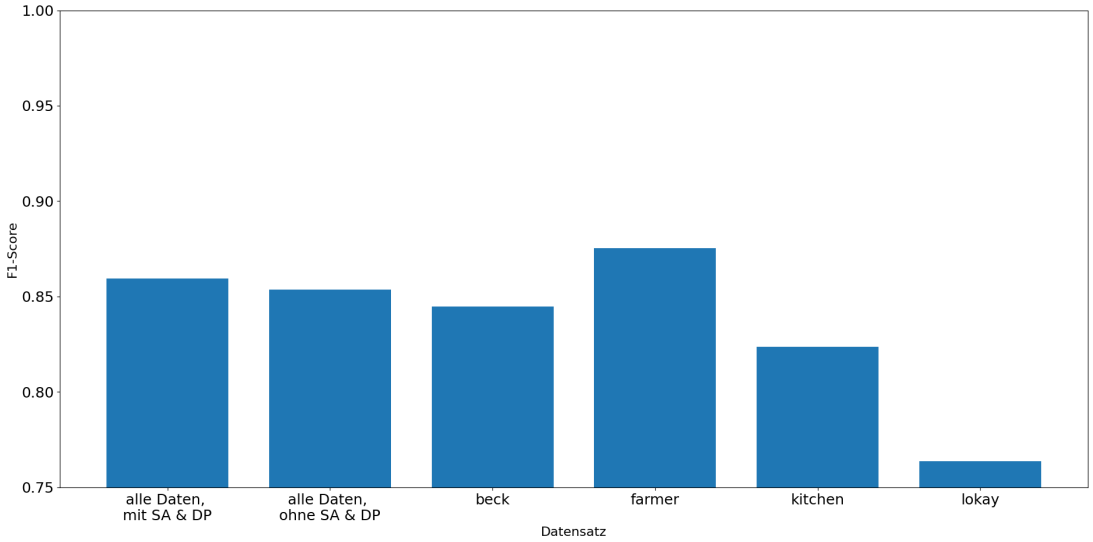

Abbildung 7.8: Effektivität (F1-Score) der Naive Bayes-Klassifikation beim zentralen Lernen mit geringer Datenmenge im Vergleich zum föderalen Lernen (Durchschnitt von fünf Durchläufen)

wie beim Szenario 4. Die Effizienz wird in diesem Szenario nicht weiter betrachtet, jedoch ist auch hier anzumerken, dass die benötigte Rechenzeit sich hier nur dadurch verändert, welche Clients beteiligt sind, da in diesem Anwendungsfall auf alle Clients gewartet wird und die Rechenzeit daher vom Client abhängt, der am längsten braucht. Die Verringerung der Clientanzahl führt noch zu einer Verkürzung der Rechenzeit bei der Koordination des föderalen Lernens, was jedoch nicht weiter betrachtet wird und auch im Verhältnis zur eigentlichen Rechenzeit der Clients vernachlässigbar ist. Die Tabelle 7.5 stellt dazu für das LSTM-Modell die durchschnittlich höchste erreichte Effektivität über fünf Durchläufe bei 30 Runden dar. Die letzte Zeile stellt zum Vergleich die Metriken für föderales Lernen mit Differential Privacy und Secure Aggregation mit allen Daten dar. Hier zeigt sich, wie zu erwarten, dass die Effektivität des LSTM-Modells mit weniger Clients abnimmt und aber von der Gesamtmenge der Trainingsdaten aller Clients, die am Training teilnehmen, abhängt. So hat die Kombination der Datensätze von kaminski, beck und williams eine Effektivität, die relativ nahe an der Effektivität mit allen Datensätzen mit sechs Clients liegt, da vor allem kaminski und beck auch die beiden größten Datensätze sind. Aus Zeitgründen bei der Auswertung wurde hier auf eine weitere Betrachtung der Effektivität beim Naive Bayes-Modell verzichtet.

Tabelle 7.5: Szenario 6: Effektivität des LSTM-Modells beim föderalen Lernen mit verringerter Clientanzahl (Durchschnitt von fünf Durchläufen)

Client	Accuracy	Precision	Recall	F1-Score
kitchen, farmer & williams	94,68%	94,75%	94,68%	94,69%
kaminski, kitchen & lokay	95,75%	95,83%	95,75%	95,73%
kaminski, beck & williams	97,86%	97,87%	97,86%	97,86%
Server (alle Daten, mit DP & SA)	98,28%	98,29%	98,28%	98,27%

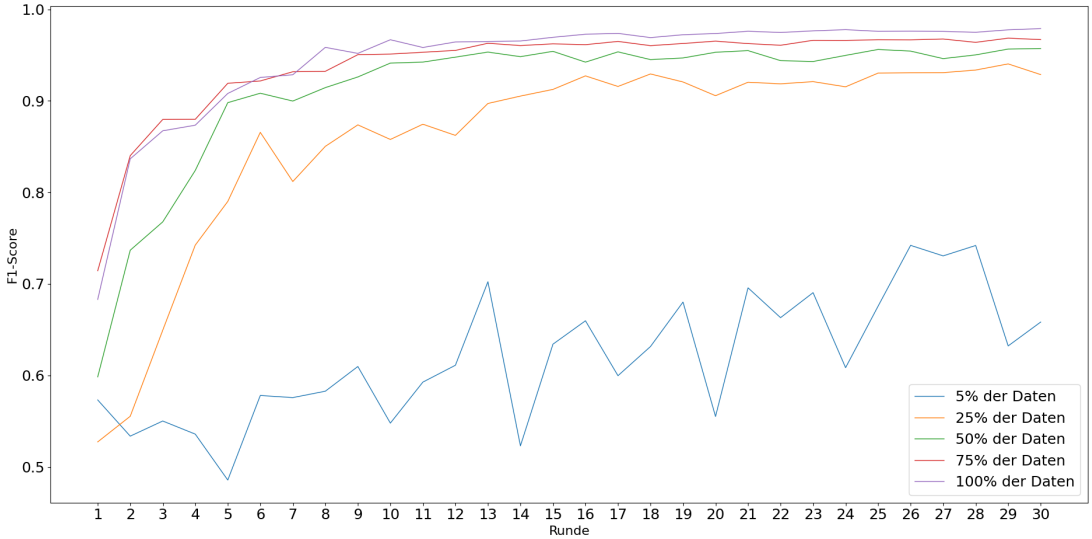

Abbildung 7.9: Effektivität (F1-Score) der LSTM-Klassifikation beim föderalen Lernen mit unterschiedlicher Datenmenge (Durchschnitt von fünf Durchläufen)

7.2.7 Szenario 7: Variation der Datenmenge beim föderalen Lernen

Bei diesem Szenario wird abschließend betrachtet, wie sich die Effektivität beim föderalen Lernen allgemein verändert, wenn die insgesamt über alle Clients verfügbare Datenmenge reduziert wird. Dazu wurde die Datenmenge für jeden Client prozentual reduziert und in Abbildung 7.9 für 30 Runden im Durchschnitt von fünf Durchläufen dargestellt. Am Verlauf der Effektivität ist sichtbar, dass die Effektivität stets mit geringerer Datenmenge auch abnimmt. Zugleich steigen die Schwankungen mit geringerer Datenmenge beim Training wie auch schon beim fünften Szenario, wo das zentrale Lernen mit geringerer Datenmenge betrachtet wurde.

8 Diskussion der Ergebnisse

In diesen Kapitel werden zuletzt die Ergebnisse aus den Szenarien des vorherigen Kapitels und die weiteren Ergebnisse der Arbeit insgesamt betrachtet. Hierbei werden die Ergebnisse beim föderalen Lernen mit den Ergebnissen beim zentralen Lernen verglichen und hinführend zum eigentlichen Ziel der Arbeit diskutiert. Dazu wird in einzelnen Abschnitten auf die verschiedenen Aspekte der Zielsetzung in den Ergebnissen eingegangen, um einen Überblick über föderales Lernen im Vergleich zum zentralen Lernen zu erhalten.

Die Umsetzung des Anwendungsfalls mit föderalem Lernen zeigt zunächst, wie eine Architektur zur Umsetzung des föderalen Lernens aussehen kann. Im Kapitel „Implementierung“ werden Details zur Umsetzung genannt und auch Herausforderungen in der Umsetzung mit den Klassifikationsverfahren beschrieben. Der Unterabschnitt 6.2 davon beschreibt die Architektur und die einzelnen Komponenten, die für das föderale Lernen notwendig sind. Hier wird vor allem auch die Kommunikation zwischen den Server und den Clients mit dem Flower Framework beschrieben. Die Grundlagen dazu wurden im Abschnitt 2.2.2 gelegt. Das Kapitel „Eingesetzte Techniken und Verfahren“ zuvor beschreibt zudem auch getroffene Entscheidungen zur Implementierung des Anwendungsfalls wie die Nutzung von Flower als grundlegende Bibliothek. Die erfolgreiche Umsetzung zeigt, dass Bibliotheken wie Flower bereits gute Möglichkeiten bieten, um Projekte mit neuronalen Netzen und föderalem Lernen umzusetzen. Implementierungen zum föderalen Lernen mit anderen Methoden als neuronale Netze wie Naive Bayes-Klassifikatoren oder Random Forests sind hingegen in den verbreiteten Bibliotheken kaum vorhanden und erschweren eine Umsetzung damit.

Im Folgenden wird die Effektivität der Klassifizierung der E-Mails beim föderalen Lernen und zentralen Lernen noch einmal allgemein betrachtet. Im Ausgangsszenario, wo keine Schutzmechanismen beim föderalen Lernen eingesetzt werden und ein zentrales Lernen zum Vergleich mit allen Daten trainiert wird, erreicht ein durch föderales Lernen trainiertes LSTM-Modell eine ähnliche Effektivität wie ein zentral trainiertes Modell. Das zentral trainierte Modell hat dabei nur einen 0,15% höheren F1-Score im Durchschnitt

(bei Precision, Recall und Accuracy verhält es sich identisch). Föderales Lernen kann somit bei LSTM-Modellen eine ähnliche Effektivität erreichen wie zentrales Lernen. Beim Naive Bayes-Modell sieht der Unterschied hingegen etwas anders aus. Dort erreicht das durch föderales Lernen trainierte Modell und durch zentrales Lernen eine erhöhte Precision im Vergleich zum Recall und zur Accuracy. Vor allem ist das zentrale Lernen hier im Vergleich zum föderalen Lernen um 2,29% effektiver im Durchschnitt. Anders als beim LSTM-Modell wurde jedoch beim Training des Naive Bayes-Modells beim föderalen Lernen eine eigene Implementierung zur Aggregation umgesetzt, da keine bestehende vorhanden war, was die Ursache für die geringere Effektivität sein kann. Insgesamt erreicht föderales Lernen hier aber abhängig vom Klassifikationsverfahren eine ähnliche bis schlechtere Effektivität bei der Klassifizierung.

Bezüglich der Effizienz gemessen an der Rechenzeit und Laufzeit ist föderales Lernen effizienter als zentrales Lernen mit allen Daten. Die Rechenzeit nur beim Training des LSTM-Modells ist bei den Messungen um 52,7% höher beim zentralen Lernen als beim föderalen Lernen und die gesamte Laufzeit ist um 8,5% höher beim zentralen Lernen. Der Unterschied ist primär bei der gesamten Laufzeit geringer, da beim föderalen Lernen Laufzeit für die Kommunikation zwischen den Clients und dem Server hinzukommt. Die Tendenz sieht beim Naive Bayes-Modell ähnlich aus, jedoch ist das Ergebnis hier nicht genau, da das Training des Naive Bayes-Modells sehr schnell ist und zudem nur eine Trainingsrunde benötigt. Im Vergleich zum Training mit allen Daten beim zentralen Lernen ist das föderale Lernen insgesamt also effizienter.

Außerdem hat das föderale Lernen im Vergleich zum zentralen Lernen keine direkte Auswirkung auf die benötigte Rechenzeit bei der Inferenz der Modelle, da die trainierten Modelle grundsätzlich identisch sind, nur mit anderen Modellparametern. Lediglich die Rechenzeit insgesamt bei der Evaluation eines Modells beim föderalen Lernen im Vergleich zum zentralen Lernen mit allen Daten ist geringer (siehe Szenario 1), da die Rechenzeit hier mit der Rechenzeit des Clients, der am längsten benötigt skaliert. Jedoch wurde hierzu in den Szenarien nicht betrachtet, wie viel länger die Evaluation durch die benötigte Kommunikation zwischen den Clients und dem Server dauert.

Beim Einsatz von Differential Privacy und Secure Aggregation sind Veränderungen bezüglich der Effektivität und Effizienz beim föderalen Lernen zu beobachten. Die Effektivität bleibt unabhängig des Klassifikationsverfahrens durch Secure Aggregation gleich, ändert sich jedoch durch Differential Privacy. Je nach gewählten Parametern zur Differential Privacy sinkt der F1-Score der trainierten Modelle deutlich. Bei der Umsetzung musste

hier jedoch eine relativ kleine Sensitivität für das Rauschen bei den LSTM-Modellen gewählt werden, was hier eventuell den Datenschutz verschlechtert und die Effektivität stärkt. Eine genauere Untersuchung zur Wahl der Parameter wäre hier weiter notwendig, um den Datenschutz und die Effektivität besser abzuwägen. Dieses Problem ist in der Umsetzung mit dem Naive Bayes-Modell jedoch nicht aufgetreten, wo die strengere (ϵ)-Differential Privacy mit dem Laplace-Mechanismus eingesetzt wurde. Hier ist auch deutlich zu erkennen, dass mit niedrigen ϵ-Werten die Effektivität sinkt. Zugleich ist hier ein überraschender Effekt beobachtet worden, da eine moderate Garantie an Differential Privacy sogar positive Effekte auf die Effektivität in den Messungen hatte und diese anstieg. Die Effizienz in Form der Rechenzeit und Laufzeit bleibt durch Differential Privacy gleich. Sie sinkt jedoch durch Secure Aggregation, da die Durchführung der Secure Aggregation jede Trainingsrunde zusätzliche Kommunikationsrunden benötigt, wodurch die Laufzeit beim LSTM-Modell um 29% steigt. Die Effizienz bei der Inferenz der Modelle bleibt unverändert, da wie beim Grundszenario an den Modellen grundsätzlich nichts verändert wird.

Im Vergleich zum zentralen Lernen mit jeweils nur einem Datensatz der Clients (ein Szenario, was realistisch ist, wenn die Daten auch nicht zentralisiert werden dürfen) hat das föderale Lernen mit Schutzmechanismen bei $\epsilon = 0,3$ eine deutlich höhere Effektivität. Hier ist beobachtbar beim LSTM-Modell, dass abhängig von der Größe des Datensatzes der Clients die Effektivität niedriger ist im Vergleich zum föderalen Lernen. Beim Naive Bayes-Modell ist die Effektivität hingegen nur auf einem ähnlichen Niveau mit Schutzmechanismen wie beim zentralen Lernen mit einem Datensatz, was aber bereits im Grundszenario beobachtet wurde und an der Implementierung der Aggregation liegen könnte.

Die Datenmenge insgesamt beim föderalen Lernen über alle Clients hat beim LSTM-Modell den zu erwartenden Einfluss auf die Effektivität der Klassifikation. Mit abnehmender Datenmenge nimmt die Effektivität immer mehr ab und das Training wird instabiler (die Effektivität schwankt zwischen den Trainingsrunden mehr). Um hier jedoch an die Effektivität des zentralen Lernens mit allen Daten im Anwendungsfall heranzukommen, muss mit allen vorhandenen Daten trainiert werden.

Durch die Variation der Clientanzahl am föderalen Lernen nimmt die Effektivität auch hier ab, da die Datenmenge insgesamt abnimmt. Je nach gewählter Kombination der Clients kommt die Effektivität dabei noch relativ nahe an die Effektivität mit allen Clients heran, wenn diese Clients etwa viele Daten in ihren Datensätzen haben oder die Daten für

das Training einen höheren Nutzen haben, da sie besser alle E-Mails repräsentieren. Auf die Effizienz hat die Clientanzahl allerdings keine direkten Einfluss, da diese weiterhin mit der Zeit skaliert, die für die Aggregation beim Server, für die Kommunikation zwischen den Clients und dem Server und der längsten benötigten Rechenzeit eines Clients benötigt wird, weil jede Runde auf alle Clients gewartet wird. Die Aggregationszeit wird zwar theoretisch durch die Clientanzahl beeinflusst, praktisch ist die Aggregationszeit jedoch so gering, dass diese vernachlässigt werden kann.

Durch das föderale Lernen wird im Vergleich zum zentralen Lernen das Netzwerk zwischen den Clients und dem Server belastet, da die Modellparameter und Metriken kommuniziert werden. In diesem Anwendungsfall fiel die Belastung dadurch jedoch eher gering aus, da es nur wenige Clients gibt und zudem die Modellparameter nicht sehr groß sind, wodurch pro Trainingsrunde jeder Client bei allen Klassifikationsverfahren höchstens 10,21 MB empfangen und 4,55 MB gesendet hat. Durch den Einsatz von Secure Aggregation verdoppelt sich diese Last pro Trainingsrunde in etwa bei jedem Client auf bis zu 19,77 MB empfangenen Daten und 10,5 MB gesendeten Daten. In Betrachtung des Anwendungsfalls mit wenigen festen Clients stellt die Auslastung des Netzwerks somit keine Herausforderung dar. Auch der Bedarf an Arbeitsspeicher während des Trainings beim föderalen Lernen wurde betrachtet, fällt jedoch mit bis zu 609 MB beim LSTM-Modell und bis zu 468 MB beim Naive Bayes-Modell eher klein aus. Auch durch den Einsatz von Secure Aggregation und Differential Privacy ändert sich dieser nicht signifikant.

In den Grundlagen der Arbeit wurde neben einer allgemeinen Vorstellung des föderalen Lernens auch betrachtet, welche Vorteile das föderale Lernen beim Datenschutz bietet und warum weitere Schutzmechanismen erfoderlich sind zum Schutz der Daten. Zentral ist dabei, dass föderales Lernen ermöglicht, dass die Eigentümer der Trainingsdaten grundsätzlich die Kontrolle über deren Verwendung durch föderales Lernen beim Training haben können, da diese nicht weitergeben werden. Somit können das Recht zur Löschung und Einschränkung der Verarbeitung, das Widerspruchsrecht, das Recht auf Vergessenwerden, Berichtigung und Datenübertragbarkeit eingehalten werden. Natürlich müssen dazu auch weiter Lösch- und Einschränkungsmöglichkeiten bei den Clients gegeben werden, um die Rechte dort zu gewährleisten. Insgesamt wird dieser Prozess vereinfacht, da die Daten nicht weitergegeben werden. Allerdings gibt es beim föderalen Lernen immer noch das Problem der Transparenz wie auch beim maschinellen Lernen, da die internen Abläufe und Entscheidungsprozesse der Modelle nicht klar offengelegt werden können. Durch die fehlende Transparenz kann auch nicht das Recht auf nicht-automatisierte Ent-

scheidungen eingehalten werden durch den Einsatz von föderalem Lernen. Zugleich muss der Prozess des föderalen Lernens durch weitere Mechanismen geschützt werden, um die Rechte vollständig einzuhalten und den Schutz zu gewährleisten, da sonst Modellparameter oder Modellausgaben Informationen über die Trainingsdaten preisgeben könnten. Hier muss gerade bei der Parameterfindung auch darauf geachtet werden, dass etwa beim Einsatz von Differential Privacy, der Schutz der Daten und die Effektivität der trainierten Modelle geeignet auf den Anwendungsfall abgewogen wird.

9 Fazit

Die Ergebnisse der Arbeit wurden im vorherigen Kapitel vorgestellt und mit den Ergebnissen beim zentralen Lernen verglichen. Im folgenden Kapitel wird nun ein Fazit gezogen, welcher die wichtigsten Erkenntnisse beim Vergleich der Ergebnisse und allgemein bei der Umsetzung zusammenfasst und hinsichtlich der Zielsetzung betrachtet.

Zentral geht aus der Umsetzung des Anwendungsfalls in dieser Arbeit hervor, dass föderales Lernen praktisch bereits mit vorhandenen Bibliotheken für neuronale Netze umgesetzt werden kann. Vor allem steht fest, dass föderales Lernen es ermöglicht, den gesetzlich geforderten Datenschutz in den meisten Punkten zu gewährleisten, sowie es zentralisiertes maschinelles Lernen nicht kann, wenn zusätzlich zum föderalen Lernen noch weitere Schutzmechanismen wie Differential Privacy und Secure Aggregation eingesetzt werden, um alle möglichen Angriffe auf das Training und die trainierten Modelle zu verhindern. Wichtig ist dabei, dass die Parameter zur Differential Privacy ausreichend für den Anwendungsfall gewählt werden, um den Datenschutz zu gewährleisten, wenn auch unter Verlust der Effektivität der Modelle. Wie die Parameter gewählt werden müssen, damit diese ausreichen, um den Datenschutz einzuhalten, wurde nicht geklärt und ist ein offenes Problem, welches weiter untersucht werden muss. Ein Training von Modellen mit föderalem Lernen, die eine ähnliche Effektivität wie zentral trainierte Modelle erreichen, ist beim Einsatz von neuronalen Netzen möglich und auch unter Einsatz von Differential Privacy kann je nach gewählten Parametern eine ausreichend hohe Effektivität für eine Klassifikation im Anwendungsfall erreicht werden. Zudem bietet föderales Lernen grundsätzlich den Vorteil im Vergleich zum zentralisierten maschinellen Lernen und kann auch zu diesem Zweck eingesetzt werden, die Systemressourcennutzung beim Training zu verteilen und so das Training zu beschleunigen, da bei festgelegten Clients die benötigte Zeit für eine Trainingsrunde primär vom Client abhängt, der am längsten braucht. So kann bei mehreren einzelnen Clients mit einer kleineren Anzahl an Daten jeweils ein Modell trainiert werden, statt zentral mit allen Daten gleichzeitig ein Modell zu trainieren. Schließlich wird auch die Dauer der Inferenz durch das föderale Lernen

nicht beeinflusst, da als Resultat gleiche Modelle entstehen, wie beim zentralen Lernen. Problematisch ist bei der Umsetzung des föderalen Lernens jedoch noch aufgefallen, dass die aktuell öffentlich zugänglichen Bibliotheken tendenziell eher experimentell sind und häufig Änderungen am Aufbau erfolgen. Auch enthalten die Bibliotheken hauptsächlich nur Implementierungen für neuronale Netze und nicht für andere Modelle. Je nach Anwendungsfall kann zudem die Netzwerklast durch das föderale Lernen für den zentralen Server ein Problem darstellen, wenn viele Clients an einer Trainingsrunde beteiligt sind, was jedoch in diesem Anwendungsfall nicht der Fall ist. Schließlich ist es noch unklar, wie auch beim föderalen Lernen Transparenz über die Verarbeitung der Daten durch die Modelle für den Datenschutz gewährleistet werden kann, was ein bestehendes Problem allgemein aus dem Bereich des maschinellen Lernens ist.

10 Ausblick

Die Durchführung, Ergebnisse und Erkenntnisse der Arbeit wurden bereits vorgestellt. In diesem letzten Kapitel erfolgt noch ein Ausblick auf mögliche Weiterentwicklungen und alternative Ansätze.

Ein Szenario, was in dieser Arbeit neben den anderen Szenarien zur Auswertung des Anwendungsfalls nicht mehr betrachtet wurde, ist, dass nicht nur die Clientanzahl beim föderalen Lernen reduziert wird, sondern zugleich auch die Datenmenge pro Client erhöht wird. Alternativ dazu könnte auch die Gesamtdatenmenge beim Vergleichsszenario mit allen Clients auf die Menge der Daten mit reduzierter Clientanzahl verringert werden. Zweck dahinter wäre, auch die direkte Auswirkung der Clientanzahl mehr bei der Aggregation zu betrachten, indem die Betrachtung nicht mehr direkt von der dadurch reduzierten Datenmenge abhängt.

Als Teil der Arbeit wurden auch Differential Privacy und Secure Multi-Party Computation (Secure Aggregation) als Schutzmechanismen betrachtet, um den Schutz der Trainingsdaten zu gewährleisten. Neben diesen beiden Optionen gibt es einige weitere Schutzmechanismen, die beim föderalen Lernen eingesetzt werden könnten und aber hier nicht weiter betrachtet wurden. Eine dieser Optionen ist die Homomorphic Encryption, mit der Berechnungen auf verschlüsselten Daten durchgeführt werden können. Diese und weitere Optionen könnten weiter im Kontext des Anwendungsfalls beim föderalen Lernen betrachtet werden, auch im Vergleich zu den bereits eingesetzten.

Des Weiteren wurde in der Arbeit nicht betrachtet, wie das Training eines Modells mit föderalem Lernen zur Klassifikation von E-Mails iterativ aussieht. In einem realen Szenario würde ein Modell regelmäßig mit neuen E-Mails aktualisiert werden, statt ein komplett neues Modell zu trainieren, was in weiteren Untersuchungen des Anwendungsfalls oder auch allgemein zum föderalen Lernen genauer betrachtet werden könnte. Dazu könnte auch die Auswirkung auf den Ansatz mit Naive Bayes zur Klassifikation betrachtet werden, indem hier statt ein Modell mit allen E-Mails gleichzeitig zu trainieren, ein Modell über mehrere Runden mit unterschiedlichen Modellen angepasst werden könnte.

Auch könnte betrachtet werden, wie ein Modell zur Mailklassifikation föderal trainiert werden könnte, wenn die Clientanzahl deutlich größer ist und zugleich die Clients nicht mehr als zuverlässig angesehen werden im Sinne des Cross-device federated learning. Sowohl beim föderalen Lernen selbst als auch etwa beim Einsatz von Secure Aggregation muss dies beachtet werden, da nicht jedes Protokoll zur Secure Aggregation gut mit den Ausfall von Clients während des Trainings umgehen kann. Daher müsste hierzu dann auch ein geeignetes Protokoll betrachtet werden, welches auch mit dem Ausfall von Clients umgehen kann.

Außerdem wurde in dieser Arbeit zur Textrepräsentation Word2Vec verwendet, was primär zwei Probleme offen lässt, die weiter untersucht werden könnten. Zum einen wird hier ein selbstrainiertes Modell verwendet, was in einer praktischen Umsetzung auch möglich wäre, jedoch müsste das Word2Vec-Modell auch föderal trainiert werden und es müsste sichergestellt werden, dass die Trainingsdaten dabei auch geschützt bleiben, da das hier verwendete Word2Vec-Modell zentral über alle Daten trainiert wurde. Zum anderen ist Word2Vec selbst nicht flexibel gegenüber unbekannten Wörtern in neuen E-Mails, wodurch diese ignoriert oder durch alternative Vektoren ersetzt werden müssten. Alternativ dazu wäre es eventuell besser, Ansätze zur Textrepräsentation zu betrachten, die dynamisch auch mit unbekannten Wörtern umgehen können, wie z.B. FastText [4] oder BERT [9] die auch mit unbekannten Wörtern eher umgehen können, indem diese weiter bis hin zu einzelnen Zeichen zerlegt werden. Auch bei anderen Ansätzen sollte dann wiederum betrachtet werden, ob ein vortrainiertes Modell verwendet werden kann oder ein eigenes Modell föderal trainiert wird.

Für die Umsetzung von Naive Bayes wurde in dieser Arbeit zur Aggregation beim föderalen Lernen lediglich der gewichtete Durchschnitt der Modellparameter gebildet. In weiteren Untersuchungen könnten auch weitere Aggregationsmethoden für den Naive Bayes-Klassifikator betrachtet werden, um eventuell ein Ergebnis zu erreichen, welches eher an ein zentral trainiertes Modell heranreicht oder dieses übertrifft. Gleiches kann auch für die Umsetzung mit einem LSTM-Modell betrachtet werden, indem auch hier andere Aggregationsmethoden untersucht werden.

Neben der Betrachtung anderer Aggregationsmethoden für Naive Bayes und LSTM könnte zudem noch betrachtet werden, wie allgemein eine erfolgreiche Aggregation von verschiedenen Random Forests beim föderalen Lernen gelingen kann. Random Forest ist hierbei als Verfahren auch interessant, da es kein Klassifikationsverfahren ist, welches

direkt greifbare Modellparameter hat, die beim föderalen Lernen aggregiert werden können.

Schließlich wurde in dieser Arbeit allgemein keine Optimierung der Parameter vorgenommen. In weiteren Betrachtungen könnten etwa die Parameter zur Differential Privacy genauer angepasst werden und dabei angemessen zwischen Effektivität und Datenschutz abgewogen werden. Hierbei wäre auch zu klären, wie überhaupt eine angemessene Abwägung zwischen Effektivität und Datenschutz vorgenommen werden kann.

Literaturverzeichnis

[1] Data Mining: Practical Machine Learning Tools and Techniques (Third Edition). In: Witten, Ian H. (Hrsg.) ; Frank, Eibe (Hrsg.) ; Hall, Mark A. (Hrsg.): *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. Third Edition. Boston : Morgan Kaufmann, 2011 (The Morgan Kaufmann Series in Data Management Systems). – URL https://www.sciencedirect.com/science/article/pii/B9780123748560000213. – ISBN 978-0-12-374856-0

[2] Bell, James H. ; Bonawitz, Kallista A. ; Gascón, Adrià ; Lepoint, Tancrède ; Raykova, Mariana: Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event USA : ACM, Oktober 2020, S. 1253–1269. – URL https://dl.acm.org/doi/10.1145/3372297.3417885. – ISBN 978-1-4503-7089-9

[3] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, August 2006. – ISBN 978-0-387-31073-2

[4] Bojanowski, Piotr ; Grave, Edouard ; Joulin, Armand ; Mikolov, Tomas: *Enriching Word Vectors with Subword Information*. Juni 2017. – URL http://arxiv.org/abs/1607.04606. – Comment: Accepted to TACL. The two first authors contributed equally

[5] Bonawitz, Keith ; Ivanov, Vladimir ; Kreuter, Ben ; Marcedone, Antonio ; McMahan, H. B. ; Patel, Sarvar ; Ramage, Daniel ; Segal, Aaron ; Seth, Karn: Practical Secure Aggregation for Privacy-Preserving Machine Learning. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA : Association for Computing Machinery, 2017 (CCS '17), S. 1175–1191. – URL https://dl.acm.org/doi/10.1145/3133956.3133982. – ISBN 978-1-4503-4946-8

[6] Bruce Guenter: *SPAM Archive*. – URL https://untroubled.org/spam/

[7] Bundesbeauftragter für Datenschutz und Informationsfreiheit: *Datenschutz-Grundverordnung - Bundesdatenschutzgesetz - Texte Und Erläuterungen*. Mai 2022. – URL https://www.bfdi.bund.de/DE/Buerger/Inhalte/Allgemein/Datenschutz/GrundlagenDatenschutzrecht.html

[8] Chevalier, Guillaume: *LARNN: Linear Attention Recurrent Neural Network*. 2018. – URL https://arxiv.org/abs/1808.05578. – Lizensiert unter CCBY-SA4.0,https://creativecommons.org/licenses/by-sa/4.0/

[9] Devlin, Jacob ; Chang, Ming-Wei ; Lee, Kenton ; Toutanova, Kristina: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Mai 2019. – URL http://arxiv.org/abs/1810.04805

[10] Dwork, Cynthia ; Roth, Aaron: The Algorithmic Foundations of Differential Privacy. In: *Foundations and Trends® in Theoretical Computer Science* 9 (2014), Nr. 3–4, S. 211–407. – URL https://www.nowpublishers.com/article/Details/TCS-042. – ISSN 1551-305X

[11] European Commission: *Horizon Europe*. Dezember 2024. – URL https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-europe_en

[12] Flower Labs GmbH: *Flower Framework Documentation*. 2024. – URL https://flower.ai/docs/framework/v1.12.0/en/index.html

[13] Fung, Benjamin C. M. ; Wang, Ke ; Chen, Rui ; Yu, Philip S.: Privacy-Preserving Data Publishing: A Survey of Recent Developments. In: *ACM Computing Surveys* 42 (2010), Juni, Nr. 4, S. 1–53. – URL https://dl.acm.org/doi/10.1145/1749603.1749605. – ISSN 0360-0300, 1557-7341

[14] Goldreich, O. ; Micali, S. ; Wigderson, A.: How to Play ANY Mental Game. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. New York, NY, USA : Association for Computing Machinery, 1987 (STOC '87), S. 218–229. – URL https://dl.acm.org/doi/10.1145/28395.28420. – ISBN 0-89791-221-7

[15] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – URL http://www.deeplearningbook.org

[16] Google: *Word2vec*. Juli 2013. – URL https://code.google.com/archive/p/word2vec/

[17] Google: *Classification: Accuracy, Recall, Precision, and Related Metrics | Machine Learning*. November 2024. – URL https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[18] Holohan, Naoise ; Braghin, Stefano ; Mac Aonghusa, Pól ; Levacher, Killian: Diffprivlib: The IBM Differential Privacy Library. In: *ArXiv e-prints* 1907.02444 [cs.CR] (2019), Juli. – URL https://github.com/IBM/differential-privacy-library

[19] Kairouz, Peter ; McMahan, H. B. ; Avent, Brendan ; Bellet, Aurélien ; Bennis, Mehdi ; Bhagoji, Arjun N. ; Bonawitz, Kallista ; Charles, Zachary ; Cormode, Graham ; Cummings, Rachel ; D'Oliveira, Rafael G. L. ; Eichner, Hubert ; Rouayheb, Salim E. ; Evans, David ; Gardner, Josh ; Garrett, Zachary ; Gascón, Adrià ; Ghazi, Badih ; Gibbons, Phillip B. ; Gruteser, Marco ; Harchaoui, Zaid ; He, Chaoyang ; He, Lie ; Huo, Zhouyuan ; Hutchinson, Ben ; Hsu, Justin ; Jaggi, Martin ; Javidi, Tara ; Joshi, Gauri ; Khodak, Mikhail ; Konečný, Jakub ; Korolova, Aleksandra ; Koushanfar, Farinaz ; Koyejo, Sanmi ; Lepoint, Tancrède ; Liu, Yang ; Mittal, Prateek ; Mohri, Mehryar ; Nock, Richard ; Özgür, Ayfer ; Pagh, Rasmus ; Raykova, Mariana ; Qi, Hang ; Ramage, Daniel ; Raskar, Ramesh ; Song, Dawn ; Song, Weikang ; Stich, Sebastian U. ; Sun, Ziteng ; Suresh, Ananda T. ; Tramèr, Florian ; Vepakomma, Praneeth ; Wang, Jianyu ; Xiong, Li ; Xu, Zheng ; Yang, Qiang ; Yu, Felix X. ; Yu, Han ; Zhao, Sen: *Advances and Open Problems in Federated Learning*. 2021. – URL https://arxiv.org/abs/1912.04977

[20] Kan, Kazutoshi: Seeking the Ideal Privacy Protection: Strengths and Limitations of Differential Privacy / Institute for Monetary and Economic Studies, Bank of Japan. URL https://ideas.repec.org/p/ime/imedps/23-e-02.html, Januar 2023 (23-E-02). – IMES Discussion Paper Series

[21] Kaspersky Lab: *Spam and Phishing in 2023*. 2024. – URL https://securelist.com/spam-phishing-report-2023/112015/

[22] Klimt, Bryan ; Yang, Yiming: *The Enron Corpus: A New Dataset for Email Classification Research*. 2015. – URL https://www.cs.cmu.edu/~enron/

[23] Li, Tian ; Sahu, Anit K. ; Talwalkar, Ameet ; Smith, Virginia: Federated Learning: Challenges, Methods, and Future Directions. In: *IEEE Signal Processing Ma-*

gazine 37 (2020), Nr. 3, S. 50–60. – URL https://ieeexplore.ieee.org/document/9084352

[24] McMahan, H. B. ; Moore, Eider ; Ramage, Daniel ; Hampson, Seth ; Arcas, Blaise A. y: *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2023. – URL https://arxiv.org/abs/1602.05629

[25] Metsis, Vangelis ; Androutsopoulos, Ion ; Paliouras, Georgios: Spam Filtering with Naive Bayes-Which Naive Bayes? In: *CEAS* Bd. 17, Mountain View, CA, 2006, S. 28–69. – URL https://pages.aueb.gr/users/ion/data/enron-spam/index.html

[26] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: *Efficient Estimation of Word Representations in Vector Space*. 2013. – URL https://arxiv.org/abs/1301.3781

[27] MUSKETEER: *MUSKETEER – Machine Learning to Augment Shared Knowledge in Federated Privacy-Preserving Scenarios*. – URL https://musketeer.eu/

[28] MUSKETEER: *Musketeer H2020 Project*. – URL https://github.com/Musketeer-H2020

[29] Olah, Christopher: *Understanding LSTM Networks*. August 2015. – URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[30] OpenMined: *PySyft*. OpenMined. Januar 2025. – URL https://github.com/OpenMined/PySyft

[31] Pedrouzo-Ulloa, Alberto ; Ramon, Jan ; Péerez-González, Fernando ; Lilova, Siyanna ; Duflot, Patrick ; Chihani, Zakaria ; Gentili, Nicola ; Ulivi, Paola ; Hoque, Mohammad A. ; Mukammel, Twaha ; Pritzker, Zeev ; Lemesle, Augustin ; Loureiro-Acuña, Jaime ; Martínez, Xavier ; Jiménez-Balsa, Gonzalo: Introducing the TRUMPET Project: TRUstworthy Multi-site Privacy Enhancing Technologies. In: *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*. Venice, Italy : IEEE, Juli 2023, S. 604–611. – URL https://ieeexplore.ieee.org/document/10224961/. – ISBN 979-8-3503-1170-9

[32] Rong, Xin: *Word2vec Parameter Learning Explained*. 2016. – URL https://arxiv.org/abs/1411.2738

[33] Roth, Holger R. ; Cheng, Yan ; Wen, Yuhong ; Yang, Isaac ; Xu, Ziyue ; Hsieh, Yuan-Ting ; Kersten, Kristopher ; Harouni, Ahmed ; Zhao, Can ; Lu, Kevin ; Zhang, Zhihong ; Li, Wenqi ; Myronenko, Andriy ; Yang, Dong ; Yang, Sean ; Rieke, Nicola ; Quraini, Abood ; Chen, Chester ; Xu, Daguang ; Ma, Nic ; Dogra, Prerna ; Flores, Mona ; Feng, Andrew: NVIDIA FLARE: Federated Learning from Simulation to Real-World. In: *IEEE Data Eng. Bull., Vol. 46, No. 1* (2023), März

[34] Russell, Stuart ; Norvig, Peter: *Artificial Intelligence, Global Edition a Modern Approach*. Pearson Deutschland, 2021. – 1168 S. – URL https://elibrary.pearson.de/book/99.150005/9781292401171. – ISBN 978-1-292-40113-3

[35] Schonlau, Matthias ; Zou, Rosie Y.: The Random Forest Algorithm for Statistical Learning. In: *The Stata Journal* 20 (2020), Nr. 1, S. 3–29. – URL https://journals.sagepub.com/doi/10.1177/1536867X20909688

[36] Sheneamer, Abdullah: Comparison of Deep and Traditional Learning Methods for Email Spam Filtering. In: *International Journal of Advanced Computer Science and Applications* 12 (2021), Nr. 1. – URL https://thesai.org/Publications/ViewPaper?Volume=12&Issue=1&Code=IJACSA&SerialNo=64

[37] Shirvani, Ghazaleh ; Ghasemshirazi, Saeid ; Beigzadeh, Behzad: *Federated Learning: Attacks, Defenses, Opportunities, and Challenges*. März 2024. – URL http://arxiv.org/abs/2403.06067

[38] Song, Congzheng ; Raghunathan, Ananth: *Information Leakage in Embedding Models*. 2020. – URL https://arxiv.org/abs/2004.00053

[39] SpamAssassin: *SpamAssassin Public Corpus*. 2006. – URL https://spamassassin.apache.org/old/publiccorpus/

[40] Truong, Nguyen ; Sun, Kai ; Wang, Siyao ; Guitton, Florian ; Guo, YiKe: Privacy Preservation in Federated Learning: An Insightful Survey from the GDPR Perspective. In: *Computers & Security* 110 (2021), S. 102402. – URL https://www.sciencedirect.com/science/article/pii/S0167404821002261. – ISSN 0167-4048

[41] Ul Haq, Ikram ; Black, Paul ; Gondal, Iqbal ; Kamruzzaman, Joarder ; Watters, Paul ; Kayes, A. S. M.: Spam Email Categorization with NLP and Using Federated Deep Learning. In: Chen, Weitong (Hrsg.) ; Yao, Lina (Hrsg.) ; Cai,

Taotao (Hrsg.) ; Pan, Shirui (Hrsg.) ; Shen, Tao (Hrsg.) ; Li, Xue (Hrsg.): *Advanced Data Mining and Applications* Bd. 13726. Cham : Springer Nature Switzerland, 2022, S. 15–27. – URL https://link.springer.com/10.1007/978-3-031-22137-8_2. – ISBN 978-3-031-22136-1 978-3-031-22137-8

[42] Unspam Technologies, Inc: *Project Honey Pot*. – URL https://www.projecthoneypot.org/index.php

[43] Yamada, Ikuya ; Asai, Akari ; Sakuma, Jin ; Shindo, Hiroyuki ; Takeda, Hideaki ; Takefuji, Yoshiyasu ; Matsumoto, Yuji: Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2020, S. 23–30

[44] Yao, Andrew C.: Protocols for Secure Computations. In: *23rd Annual Symposium on Foundations of Computer Science (Sfcs 1982)*, URL https://ieeexplore.ieee.org/document/4568388, 1982, S. 160–164

A Anhang

A.1 Verwendete Hilfsmittel

In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Bachelorarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	Verwendung
ChatGPT	LLM, verwendet zur Recherche von Grundliteratur und Erzeugung von Erklärungen zum eigenen Verständnis
draw.io	Verwendet zur Gestaltung eigener Diagramme
LaTeX	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments
Visual Studio Code	Verwendet zur LaTeX-Bearbeitung der Arbeit und zur Bearbeitung des Programmcodes
Zotero	Eingesetzt zum Sammeln von Quellen und zur Erzeugung der BibTeX-Quellenangaben

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original