HAW HAMBURG

Masterarbeit

Burak Tinman

Monolinguale Maschinenübersetzung von Standardsprache in Leichte Sprache mittels Retrieval-Augmented Generation und Large Language Models

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Burak Tinman

Monolinguale Maschinenübersetzung von Standardsprache in Leichte Sprache mittels Retrieval-Augmented Generation und Large Language Models

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Martin Schultz

Eingereicht am: 21.08.2025

Burak Tinman

Thema der Arbeit

Monolinguale Maschinenübersetzung von Standardsprache in Leichte Sprache mittels Retrieval-Augmented Generation und Large Language Models

Stichworte

Maschinenübersetzung, Leichte Sprache, Large Language Models, Retrieval-Augmented Generation, In-context Learning, Fine-Tuning

Kurzzusammenfassung

Barrierefreiheit und Inklusion erfordern zugängliche Informationen. Insbesondere im öffentlichen Sektor ist dabei die Leichte Sprache von großer Bedeutung. Manuelle Übersetzungen sind jedoch zeit- und kostenintensiv. Diese Arbeit untersucht deshalb die automatisierte, monolinguale Textvereinfachung von deutscher Standardsprache in Leichte Sprache, mit einem spezifischen Fokus auf die Anforderungen des öffentlichen Sektors. Im Mittelpunkt steht der systematische Vergleich der Leistungsfähigkeit kleinerer Large Language Models mit jener großer, dem Stand der Technik entsprechender Modelle. Der Vergleich berücksichtigt neben qualitativen Anforderungen, evaluiert anhand der Kriterien Einfachheit, Grammatikalität und Semantik durch automatisierte Metriken, auch wirtschaftliche und rechtliche Aspekte, die im öffentlichen Sektor ebenfalls relevant sind. Darüber hinaus wird die Wirksamkeit der zwei Optimierungsstrategien In-context Learning mittels Retrieval-Augmented Generation und Fine-Tuning der Modelle evaluiert. Die Ergebnisse belegen, dass State-of-the-Art-Modelle zwar gute Übersetzungen liefern, aber optimierte kleinere Sprachmodelle eine kompetitive oder sogar überlegene Leistung erzielen können. Retrieval-Augmented Generation verbessert die Textqualität durch die Einbeziehung domänenspezifischer Beispiele, und auch das Fine-Tuning erweist sich als robuste Methode zur Leistungssteigerung. Schließlich wird gezeigt, dass optimierte kleinere Sprachmodelle eine leistungsstarke, kosteneffiziente und datenschutzkonforme Alternative zu großen, cloudbasierten Modellen darstellen. Sie bieten öffentlichen Institutionen eine praktikable Lösung, um die qualitativen, wirtschaftlichen und gesetzlichen Anforderungen an die digitale Barrierefreiheit zu erfüllen.

Burak Tinman

Title of Thesis

Monolingual Machine Translation from Standard Language to Easy Language using Retrieval-Augmented Generation and Large Language Models

Keywords

Machine Translation, Easy Language, Large Language Models, Retrieval-Augmented Generation, In-context Learning, Fine-Tuning

Abstract

Accessibility and inclusion require accessible information. In the public sector in particular, Easy Language is of great importance. However, manual translations are time-consuming and costly. This work therefore investigates the automated, monolingual text simplification from standard German into Easy Language, with a specific focus on the requirements of the public sector. The core of the study is a systematic comparison of the performance of smaller Large Language Models with that of large, state-of-the-art models. The comparison considers not only qualitative requirements, evaluated using automated metrics based on the criteria of simplicity, grammaticality, and semantics, but also economic and legal aspects relevant to the public sector. Furthermore, the effectiveness of the two optimization strategies In-context Learning via Retrieval-Augmented Generation and Fine-Tuning of the models is evaluated. The results demonstrate that while state-of-the-art models already provide good translations, optimized smaller language models can achieve competitive or even superior performance. Retrieval-Augmented Generation improves text quality by incorporating domain-specific examples, and Fine-Tuning also proves to be a robust method for performance enhancement. Finally, it is shown that optimized smaller language models represent a high-performing, cost-effective, and privacy-compliant alternative to large, cloud-based models. They offer public institutions a viable solution to meet the qualitative, economic, and legal requirements for digital accessibility.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADÜ Nord Assoziierten Dolmetscher und Übersetzer in Norddeutschland.

BDSG Bundesdatenschutzgesetz.

BDÜ Bundesverband der Dolmetscher und Übersetzer.

BERT Bidirectional Encoder Representations from Transformers.

BGG Behindertengleichstellungsgesetz.

BITV Barrierefreie-Informationstechnik-Verordnung.

BLEU Bilingual Evaluation Understudy.

BPE Byte-Pair-Encoding.

CAT Computer-assisted Translation.

CBOW Continuous Bag of Words.

DSGVO Datenschutz-Grundverordnung.

ICL In-context Learning.

ILSMH International League of Societies for Persons with Mental Handicap.

IR Information Retrieval.

KI Künstliche Intelligenz.

LHA Large-Scale Hierarchical Alignment.

LLMs Large Language Models.

LoRA Low-Rank Adaptation.

LSTM Long Short-Term Memory.

NLP Natural Language Processing.

PEFT Parameter-efficient Fine-Tuning.

QLoRA Quantized Low-Rank Adaptation.

RAG Retrieval-Augmented Generation.

RLHF Reinforcement Learning from Human Feedback.

RNN Rekurrente Neuronale Netzwerke.

SARI System output Against References and Input.

SLMs Small Language Models.

SOTA State of the Art.

TM Translation Memory.

TU Translation Unit.

UN-BRK UN-Behindertenrechtskonvention.

Listings

1 Einleitung

1.1 Motivation

Um den Anforderungen von Barrierefreiheit und Inklusion in der digitalen Kommunikation gerecht zu werden und Informationen einem breiten Publikum zugänglich zu machen, ist der Einsatz von Leichter Sprache ein vielversprechender Weg. Diese zeichnet sich als eine vereinfachte Varietät des Deutschen mit reduziertem Satzbau, einem limitierten Wortschatz, einem geringeren Maß an vorausgesetztem Weltwissen und einer klaren visuellen Strukturierung aus [92, S. 11-12]. Insbesondere Personen mit kognitiven Einschränkungen, mangelnden Lese- und Schreibfertigkeiten oder Deutsch als Zweitsprache profitieren von dieser sprachlichen Aufbereitung durch ein verbessertes Textverständnis, wobei auch für nicht-beeinträchtigte Personen positive Effekte nachgewiesen wurden [131]. Die Relevanz dieser sprachlichen Anpassung wird durch Befunde wie die der LEO-Studie 2018 [60, S. 20] unterstrichen, die einen Anteil von etwa 6,2 Millionen Erwachsenen in Deutschland mit geringer Literalität identifiziert. Weitere Schätzungen deuten darauf hin, dass das Potenzial der Zielgruppe für Leichte Sprache zwischen 14 und 16 Millionen Personen umfasst und dieser Personenkreis ebenfalls von Leichter Sprache profitieren kann [29, 41].

Eine Notwendigkeit zur Umsetzung der Barrierefreiheit und Inklusion in der digitalen Kommunikation ist mit der Barrierefreie-Informationstechnik-Verordnung (BITV) [26] und dem Behindertengleichstellungsgesetz (BGG) [25] für Informationen im Web im öffentlichen Sektor zudem rechtlich gegeben. Diese Gesetzgebungen wurden u.a. auf Basis der EU-Richtlinie 2016/2102 [48] über den barrierefreien Zugang zu Webangeboten öffentlicher Stellen maßgeblich beeinflusst. Die EU-Richtlinie verpflichtet die Mitgliedstaaten dazu, dass öffentliche Stellen von der Bundes- bis zur kommunalen Ebene barrierefreie Webangebote bereitstellen. Entsprechend wird die Handhabung und Umsetzung der Leichten Sprache in Deutschland im öffentlichen Sektor insbesondere durch § 4 BITV und § 11 BGG geregelt.

Um den Zugang zu Informationen an öffentlichen Stellen zu verbessern, nutzen z.B. Behörden mit Webangeboten Übersetzungsdienste, bei denen komplexe Artikel in Leichte Sprache übersetzt werden. Während einige Behörden automatisierte Lösungen einsetzen, stützt sich die Mehrheit auf manuelle Übersetzungsprozesse [7]. Eine manuelle Vereinfachung ist jedoch nicht nur zeitaufwendig und kostspielig, da sie geschulte Fachkräfte erfordert, sondern auch ineffizient, um mit der ständig wachsenden Menge an Online-Informationen umzugehen. Hierbei können (semi-)automatisierte Lösungen bei Übersetzungen unterstützen [47, 144]. Daher besteht ein gesellschaftlicher Bedarf an automatisierten oder zumindest halbautomatischen Textvereinfachungsverfahren, die durch den Einsatz von Natural Language Processing (NLP) eine schnellere, skalierbare und flexiblere Lösung für die Erstellung kognitiv zugänglicher Texte ermöglichen [47].

Eine Möglichkeit ist die Nutzung von Large Language Models (LLMs), die einen vielversprechenden Weg bieten, diesen manuellen Aufwand durch automatisierte maschinelle Übersetzung zu reduzieren. LLMs lassen sich auf eine Vielzahl von NLP-Aufgabenfeldern anwenden [102] und können zudem für Maschinenübersetzungen und Textvereinfachungen eingesetzt werden [2, 16, 99].

Die Nutzung von LLMs mit hoher Parameteranzahl bzw. sogenannten State of the Art (SOTA)- oder proprietären Modellen, insbesondere von Anbietern wie OpenAI, Google oder Anthropic, birgt jedoch Risiken in Bezug auf Datenschutz und Privatsphäre bei der Verarbeitung von Daten [61, 132, 134] sowie hinsichtlich der Nutzungskosten in Bezug auf Hardware und Inferenz [127, 129, 160].
In Deutschland gilt zudem die Datenschutz-Grundverordnung (DSGVO) auf Basis der EU-Verordnung 2016/679 [49], die durch das Bundesdatenschutzgesetz (BDSG) [24] ergänzt wird und Angelegenheiten zu Datenschutz und Privatsphäre regelt. Somit muss die Nutzung von LLMs auch diesen rechtlichen Rahmenbedingungen folgen, vor allem im Bereich des öffentlichen Sektors.
In Bezug auf die Kosten bei der Nutzung von LLMs bzw. SOTA-Modellen lässt sich feststellen, dass vor allem LLMs mit hoher Parameteranzahl aufgrund ihres höheren Rechenbedarfs mehr Kosten verursachen als kleinere Modelle [129]. Dies kann in einem kostenbeschränkten Umfeld wie dem öffentlichen Sektor eine entscheidende Rolle bei der Entscheidung zur Nutzung von LLMs oder anderen automatischen Textvereinfachungsverfahren gegenüber manueller Übersetzungsdienstleistungen spielen.

Auf Grundlage der genannten qualitativen, wirtschaftlichen und rechtlichen Anforderungen zur Bereitstellung der Leichten Sprache in öffentlichen Stellen ist das Ziel dieser

Arbeit daher, die Fähigkeiten von LLMs zur automatischen Erstellung von Texten in Leichter Sprache aus der Standardsprache zu untersuchen. Dies soll unter der Betrachtung der Bedingungen speziell für den öffentlichen Sektor geschehen, mit der Prüfung von Optimierungspotenzialen bei der Nutzung von kleineren Sprachmodellen durch In-context Learning (ICL) mittels Retrieval-Augmented Generation (RAG) sowie durch das Fine-Tuning der Modelle.

1.2 Forschungsfragen

Aufbauend auf der dargelegten Motivation, die den Bedarf an Leichter Sprache im öffentlichen Sektor, die Herausforderungen manueller Übersetzungsprozesse und die Potenziale von LLMs zur automatischen Textvereinfachung aufzeigt, fokussiert sich diese Arbeit auf die folgenden Forschungsfragen:

1. **Inwieweit sind LLMs in der Lage, automatisiert Texte in Leichter Sprache für den öffentlichen Sektor zu erstellen, die den Kriterien zur Bewertung der Leichten Sprache hinsichtlich Einfachheit, Grammatikalität und Semantik entsprechen?**
   Diese Frage untersucht die grundlegende Fähigkeit von LLMs, Texte zu erstellen, die der Leichten Sprache im öffentlichen Sektor genügen. Dabei werden Aspekte wie die Einfachheit des Zieltexts gegenüber dem Ausgangstext (Einfachheit), die grammatikalische Korrektheit sowie allgemeine Verständlichkeit (Grammatikalität) und die Beibehaltung der Bedeutung des Textes (Semantik) mithilfe automatisierter Metriken analysiert. In dieser Frage soll vor allem die Übersetzungsqualität von LLMs ohne weitere Anpassungen durch Optimierungsstrategien überprüft werden.

2. **Können die Optimierungsstrategien ICL mittels RAG und Fine-Tuning die Leistung von LLMs bei der automatischen Erstellung von Texten in Leichter Sprache im öffentlichen Sektor verbessern?**
   Diese Frage zielt darauf ab, die Optimierungsstrategien ICL mittels RAG und Fine-Tuning in Bezug auf Leichte Sprache im öffentlichen Sektor zu bewerten und sie anhand der Aspekte der Einfachheit, Grammatikalität und Semantik im Vergleich zu den Ausgaben der LLMs ohne Nutzung der Optimierungsstrategien zu vergleichen. Dabei wird auch die Kombination beider Optimierungsstrategien hinsichtlich ihrer Auswirkungen auf die Qualität der erstellten Texte untersucht.

3. **Inwieweit lassen sich LLMs mit kleinerer Parameterzahl im Vergleich zu LLMs mit hoher Parameterzahl bzw. SOTA-Modellen effektiv für die automatische Erstellung von Leichter Sprache im öffentlichen Sektor einsetzen, unter Berücksichtigung von qualitativen, wirtschaftlichen und rechtlichen Anforderungen an die Übersetzungen?**
   Zur Beantwortung dieser Frage sollen kleine Sprachmodelle mit aktuellen großen Sprachmodellen im Kontext der Anforderungen des öffentlichen Sektors verglichen werden. Dabei werden vor allem die qualitativen und wirtschaftlichen Aspekte berücksichtigt. Zudem sollen rechtliche Anforderungen, soweit es möglich ist, ebenfalls beleuchtet werden, um eine umfassende Bewertung zu erhalten.

Durch die Beantwortung dieser Fragen soll ein Beitrag zur Entwicklung von automatisierten Lösungen für die Textvereinfachung in Leichte Sprache geleistet werden, die speziell auf die Bedürfnisse und Rahmenbedingungen des öffentlichen Sektors zugeschnitten sind.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in insgesamt fünf Kapitel, wobei nach der Einleitung zunächst grundlegende Konzepte beschrieben werden. Anschließend wird die technische Umsetzung erläutert sowie die resultierenden Ergebnisse interpretiert und eingeordnet. Die Arbeit wird mit einem abschließenden Fazit abgeschlossen.

Kapitel 2 (Grundlagen) legt die theoretische Basis für das Verständnis der Arbeit dar, wobei zentrale Konzepte der Leichten Sprache, der monolingualen Maschinenübersetzung sowie von LLMs detailliert erläutert werden. Zusätzlich wird auf den aktuellen Forschungsstand im Bereich der Maschinenübersetzungen in Leichte Sprache eingegangen. Dadurch wird ein Fundament geschaffen, um die methodischen Entscheidungen und Forschungsergebnisse im weiteren Verlauf nachvollziehen zu können.

Den Kern der Arbeit bildet die technische Umsetzung in **Kapitel 3 (Konzeption und Implementation)**. In diesem Kapitel wird der gesamte Prozess von Konzeption sowie Implementierung der Lösungen detailliert beschrieben. Dies umfasst die Aufbereitung der Datensätze, die Auswahl der Sprachmodelle, die Implementierung des RAG-Systems sowie das Fine-Tuning der Modelle. Abschließend wird der Aufbau der Evaluation sowie deren Ergebnisse dargelegt.

In **Kapitel 4 (Diskussion)** werden die Ergebnisse interpretiert und diskutiert. Die Resultate werden dabei im Kontext der formulierten Forschungsfragen analysiert. Zudem erfolgt eine Einordnung der Ergebnisse in den breiteren Forschungskontext und eine Auseinandersetzung mit den Limitationen der durchgeführten Untersuchung.

Abschließend fasst **Kapitel 5 (Fazit)** die Erkenntnisse der Arbeit zusammen und beantwortet die Forschungsfragen. Darauf aufbauend wird ein Ausblick auf mögliche weiterführende Forschungsarbeiten und zukünftige Potenziale in diesem Bereich gegeben.

2 Grundlagen

2.1 Leichte Sprache

2.1.1 Definition und Regeln von Leichter Sprache

Nach Maaß und Rink lässt sich Leichte Sprache folgendermaßen definieren:

> „Leichte Sprache ist eine Varietät des Deutschen, die gegenüber dem voll ausgebauten Standarddeutschen eine erhöhte Wahrnehmbarkeit und Verständlichkeit für Personen mit Leseeinschränkungen aufweist.“ [94, S. 251]

Grundlegende Eigenschaften der Leichten Sprache bilden dabei, wie bereits geschildert, ein reduzierter Satzbau, Wortschatz und vorausgesetztes Weltwissen sowie eine besondere visuelle Aufbereitung [92, S. 11-12].

Zu unterscheiden ist die Leichte Sprache von der Einfachen Sprache, die sich durch einen „komplexeren Sprachstil“ [79] auszeichnet. Kellermann unterscheidet Leichte Sprache und Einfache Sprache textuell anhand einiger Merkmale [79], die in Tabelle 2.1 dargestellt sind. So folge die Leichte Sprache strengen und klar definierten Regeln, nutze ausschließlich kurze Hauptsätze und vermeidet komplexe Wörter. Als zu schwierig bezeichnete Wörter sollen bestmöglich erklärt und Fremdwörter generell vermieden werden. Die Schrift solle serifenlos und klar sein, der Text durch Absätze gegliedert und mit wenigen Farbakzenten versehen werden. Im Vergleich dazu existiere bei der Einfachen Sprache keine strenge Regulierung, was sie flexibler bei der Ausgestaltung der Texte mache.

Die streng und klar definierten Regeln der Leichten Sprache können aus Regelwerken entnommen werden, z.B. aus den nicht-rechtlich bindenden Regelwerken des Netzwerks Leichte Sprache [111] sowie des Regelwerks von Inclusion Europe [74]. Darüber hinaus enthält die für die Bundesverwaltung rechtlich bindende BITV entsprechende Regeln, welche im Anhang der Verordnung zu finden sind [72].

Merkmal	Leichte Sprache	Einfache Sprache
Regeln	Sehr strenge, klar definierte Regeln.	Keine strengen Regeln; flexibler.
Satzbau	Kurze Hauptsätze; vermeidet Nebensätze.	Längere Sätze erlaubt; Nebensätze zulässig.
Vokabular	Bekannte Wörter; schwierige werden erklärt.	Alltagsbegriffe vorausgesetzt; Fremdwörter vermeiden/erklären.
Optik	Klare, serifenlose Schrift; Absätze nach Satzzeichen; sparsame Farbe.	Weniger strenge Anforderungen; Fokus auf Lesbarkeit.
Fremdwörter	Generell vermeiden.	Möglichst vermeiden; erklären.

Tabelle 2.1: Vergleich von Leichter Sprache und Einfacher Sprache (vgl. [79])

Diese Regelwerke beschreiben u.a., welche Wörter, Zahlen, Zeichen oder Sätze genutzt werden sollten, welche Gestalt der Text haben soll, und sie geben Prüfern weitere Hinweise zur Umsetzung und Überprüfung. In allen drei Regelwerken lassen sich insgesamt 17 Regeln finden, die „konvergieren“ [94, S. 253], d.h., die Regeln stimmen trotz unterschiedlicher Formulierungen inhaltlich überein. In Tabelle 2.2 sind die konvergierenden Regeln einsehbar.

Trotz der vorhandenen Regelwerke sind sie lediglich „dazu geeignet, Texte in Leichte Sprache zu identifizieren“. Aber „[f]ür eine professionelle Textpraxis sind sie jedoch nicht differenziert genug, da sie nur einen gewünschten Zustand beschreiben („Kurze Sätze“), aber keinen Ansatz dafür enthalten, wie dieser Zustand erreicht werden kann“ [94, S. 253]. Beispielsweise enthält die Anlage 2 (zu § 3 Absatz 2) des BITV 13 Vorgaben, ohne konkrete Beispiele oder Umsetzungsmöglichkeiten zu nennen oder zu beschreiben [27]. Außerdem entsprechen die vorliegenden Regelwerke keinem Standard und sind auch nicht normiert.

Nach Husel zielt Leichte Sprache auf größtmögliche Verständlichkeit ab. Jedoch können die strengen Regeln und Vereinfachungen unbeabsichtigt zu Problemen führen. Eine vereinfachte Grammatik oder ein auffälliges Layout kann unter Umständen auf geringe Akzeptanz in der Öffentlichkeit stoßen und aufgrund der deutlichen Unterscheidung zur Standardsprache die Nutzer der Leichten Sprache stigmatisieren. Eine dazu vorgeschlagene weitere Varietät, um diese Herausforderungen zu mitigieren, könne der von Maaß vorgestellte Vorschlag Leichte Sprache Plus sein, die darauf abziele, die negativen Aspekte zu vermeiden [72, S. 28].

Kategorie	Regeln
Visuelle und mediale Gestaltung	1. Größere Schriftgröße 2. Jeder Satz auf eine neue Zeile 3. Keine Worttrennung am Zeilenende 4. Linksbündig
Morphologie	5. Kurze Wörter 6. Trennung von komplexen Wörtern durch Bindestriche 7. Verbot von Abkürzungen 8. Passiv vermeiden
Lexik	9. Leicht verständliche Wörter 10. Möglichst keine Fremdwörter 11. Fremdwörter erklären
Syntax	12. Kurze Sätze
Semantik	13. Negation vermeiden
Text	14. Konsistenz in der Bezeichnung auf Ebene der Substantive 15. Relevante Informationen an den Anfang 16. Zwischenüberschriften erwünscht 17. Direkte Ansprache

Tabelle 2.2: Konvergierende Regeln von Netzwerk Leichte Sprache, Inclusion Europe und BITV 2.0 aus [94, S. 253] nach [19]

Sie „stellt den Kipppunkt der Balance zwischen den vier Merkmalen Wahrnehmbarkeit, Verständlichkeit, Akzeptanz und Stigmatisierungsrisiko“ dar [93, S. 232]. Nach Rink und Maaß lässt sich daher Leichte Sprache Plus zwischen der Leichten Sprache und der Einfachen Sprache als „Bindeglied“ einordnen. Zur Erreichung einer nicht-stigmatisierenden und nutzerfreundlichen Darstellung, die gleichzeitig informativ ist, werden die Regeln der Leichten Sprache als Ausgangspunkt genommen und gezielt ergänzt. Das geschieht durch einen größeren Wortschatz, die Einführung von Fachbegriffen und einfache Nebensätze, während die Information im Text verdichtet wird [124, S. 495-496].

Leichte Sprache zeichnet sich also durch klar definierte Regeln aus, die auf einen reduzierten Satzbau, Wortschatz und eine besondere visuelle Aufbereitung abzielen, um die Verständlichkeit für Menschen mit Leseeinschränkungen zu erhöhen.

2.1.2 Geschichtliche und rechtliche Hintergründe von Leichter Sprache

Die konzeptuellen Anfänge der Leichten Sprache liegen in Schweden im Jahre 1968, als das Komitee der Swedish National Agency for Education den Ansatz aufnahm. Zwischen

den 1960ern und 1980ern wurden dann erste Bücher und Zeitungen in einfacher schwedischer Sprache („Lättläst“) herausgebracht [6, 79].
Parallel entstand im Jahre 1974 die People-First Organisation, welche zur Stärkung der Rechte von Menschen mit Behinderungen gegründet wurde und die „Easy Read“-Initiative ins Leben rief, wodurch auch im angloamerikanischen Bereich die ersten Regeln zur leichteren Verständlichkeit von Texten entstanden [50, 79] (siehe auch [11, S. 73]).
In Deutschland wurde durch das Projekt „Wir vertreten uns selbst!“ des Netzwerks von Menschen mit Lernschwierigkeiten die Idee der Leichten Sprache erstmals 1997 vorangetrieben, welches von der Mensch zuerst - Netzwerk People First Deutschland e.V. 2001 auch offiziell als Verein aufgenommen wurde. Neben verschiedenen Projekten zwischen 1997 und 2012 mit dem Bundesministerium für Gesundheit und Soziales, brachte der Verein zwei Wörterbücher im Jahre 2000 und 2008 in Leichter Sprache heraus [112].
Im Jahre 2006 entstand das Netzwerk Leichte Sprache in Deutschland [110], das 2009 parallel zur Organisation Inclusion Europe ein umfassendes Regelwerk zur Leichten Sprache herausbrachte. Dieses wurde zusätzlich durch Personen aus der Zielgruppe überprüft [74, 111].

Auf Basis der Entstehungsgeschichte der Leichten Sprache und der generellen Bürgerrechtsbewegungen seit 1960, wurde die rechtliche Verankerung von Behindertenrechten, einschließlich der Forderung nach Leichter Sprache, verfolgt.
Die europäische Vereinigung International League of Societies for Persons with Mental Handicap (ILSMH) entwickelte und gab 1998 „erstmals Europäische Richtlinien für die Erstellung von leicht lesbaren Informationen“ heraus [79]. Diese Richtlinien [52] dienten als Basis für die Erstellung des Regelwerks von Inclusion Europe.
Im Gegensatz zur ILSMH war die Verabschiedung der UN-Behindertenrechtskonvention (UN-BRK) 2006 [28] offiziellerer Natur und hatte maßgeblichen Einfluss auf die Gesetzgebungen weltweit, darunter auch zur barrierefreien Kommunikation [94, S. 78-81]. Nach der Ratifikation im Dezember 2008 im Bundestag trat die UN-BRK im Januar 2009 in Deutschland in Kraft [30] und führte zur Novellierung des BGG [25], die den öffentlichen Sektor dazu verpflichtete, Informationen und Bescheide in Leichter Sprache zur Verfügung zu stellen und zu erläutern [38]. Mit der EU-Richtlinie 2016/2102 [48] wurde eine weitere Novelle des BGG angestoßen und mit der BITV [26] 2018 in Deutschland umgesetzt, bei der auch barrierefreie Webangebote, u.a. durch Informationen in Leichter Sprache, von Bundes- bis auf kommunaler Ebene erforderlich sind. Die Regelungen zur Leichten Sprache in Deutschland werden dabei durch § 4 BITV und § 11 BGG bestimmt.

Ausgehend von diesen rechtlichen Rahmenbedingungen sind die Behörden in Deutschland dazu verpflichtet, ihre Informationen in Leichter Sprache anzubieten

2.1.3 Übersetzung von Standardsprache in Leichte Sprache

Die Übersetzung der deutschen Standardsprache in Leichte Sprache lässt sich als eine intralinguale Übersetzung bezeichnen, da „von fachlicher Varietät [...] keine Sprachgrenze, sondern eine Varietätengrenze innerhalb einer Einzelsprache überschritten“ werde [19, Kapitel 6.2]. Dies stütze sich nach Bredel auf die Annahme von Zethsen, dass:

> „A source text exists or has existed at some point in time. A transfer has taken place and the target text has been derived from the source text (resulting in a new product in another language, genre or medium), i. e. some kind of relevant similarity exists between the source and the target texts. This relationship can take many forms and by no means rests on the concept of equivalence, but rather on the skopos of the target text.“ [173]

Zur Umsetzung der Übersetzung spielen nach Husel verschiedene Akteure mit unterschiedlichen Mitarbeiterprofilen eine Rolle [72, S. 59-61]. Husel stellte in diesem Zusammenhang das Modell nach Maaß vor, das die Profile in „Text creators“, „Text users“ und „Bystanders“ unterscheidet [93, S. 169-171]:
„Text creators“ sind dabei für die Produktion der Texte in Leichter Sprache verantwortlich und nehmen die Rolle als Übersetzer oder u.U. auch als Überprüfer der Texte ein, die aus der primären Zielgruppe stammen. Mit „Text users“ sind Fachexperten oder Verwaltungsexperten gemeint, die die Produkte aus der Übersetzung zur Kommunikation mit der primären Zielgruppe nutzen. Abschließend wird mit dem Begriff „Bystander“ „sekundäre Adressaten“ beschrieben, die das Angebot der Leichten Sprache zwar nicht nutzen müssen, jedoch wahrnehmen.
Die vier Hauptprofile der „Text creators“ nach Maaß umfassen solche mit starkem Zielgruppenbezug aus dem Empowerment-Bereich, professionelle Übersetzer mit Zusatzqualifikation in Leichter Sprache, akademisch ausgebildete Übersetzer für Leichte Sprache sowie spezialisierte Redakteure in Medieninstitutionen [93, S. 177-178]. Zusätzlich müssen diese Experten über vier Anforderungen verfügen: Fach- und Fachsprachenkompetenz, Kenntnis der Zielgruppe und Zielsituation, Kenntnis Leichter Sprache und ihrer Dilemmata und Translationskompetenz [72, S. 67] (siehe auch [93, S. 172-173]).

Im Jahr 2016 waren Übersetzer der Leichten Sprache laut Bredel noch kein Bestandteil von Berufsgemeinschaften, wie dem Bundesverband der Dolmetscher und Übersetzer (BDÜ) sowie dem Assoziierten Dolmetscher und Übersetzer in Norddeutschland (ADÜ Nord) [19, Kapitel 6.3]. Seit 2018 sind sie jedoch in beiden Berufsgemeinschaften eingegliedert. Dabei bietet die BDÜ Zertifizierungsprogramme zur Ergänzung ihres Tätigkeitsfelds um die Leichte Sprache. Auch in universitären Ausbildungsorten ist Leichte Sprache ein Thema [94, S. 277-278]. Übersetzungsleistungen können nach Bredel zusätzlich unterstützt werden durch „Wörterbücher für Leichte Sprache“, „Inhaltsparadigmatische Wörterbücher des Deutschen“, „Tools zur Verständlichkeitsprüfung“, „Terminologiemanagementsysteme“ oder auch von „Translation Memory (TM)“-Systemen [19, Kapitel 6.7.2]. Auch Maaß nennt den Einsatz von TM-Systemen als eine solche Unterstützungsmöglichkeit [94, S. 588].

Um die linguistischen Unterschiede zwischen der Standardsprache und der Leichten Sprache zu demonstrieren, werden im Folgenden jeweils die ersten drei Regeln aus dem Regelwerk [111, S. 11-32] von Netzwerk Leichte Sprache e.V. beispielhaft für die Themen Wörter, Zahlen und Zeichen sowie Sätze dargestellt.

Wörter

- **W1. Benutzen Sie einfache Wörter**:
  Statt des Wortes „genehmigen“, solle das Wort „erlauben“ benutzt werden.
- **W2. Benutzen Sie Wörter, die etwas genau beschreiben**:
  Statt „Öffentlicher Nahverkehr“, solle beispielsweise „Bus und Bahn“ genutzt werden
- **W3. Benutzen Sie bekannte Wörter. Verzichten Sie auf Fachwörter und Fremdwörter**:
  Statt „Workshop“, solle das Wort „Arbeits-Gruppe“ benutzt werden.
  Die Erklärung eines schweren Wortes soll vorher angekündigt werden und wird anhand des folgenden Beispiels dargelegt:

  > „Herr Meier hatte einen schweren Unfall.
  > Jetzt lernt er einen anderen Beruf.
  > Das schwere Wort dafür ist:
  > berufliche Rehabilitation.“

Zahlen und Zeichen

- **Z1. Schreiben Sie Zahlen so, wie die meisten Menschen sie kennen**:
  Statt römische Zahlen zu nutzen wie „IX“, solle die arabische Zahl „9“ genutzt werden.
- **Z2. Vermeiden Sie alte Jahres-Zahlen**:
  Beispielhaft wird statt der Nutzung von „1867“ die Nutzung von „Vor langer Zeit“ oder „Vor mehr als 100 Jahren“ empfohlen.
- **Z3. Vermeiden Sie hohe Zahlen und Prozent-Zahlen**:
  Hohe Zahlen oder Prozentzahlen sollen entweder durch Vergleiche oder ungenaue Angaben übersetzt werden. Beispielsweise wird aus „14.795 Menschen“ „Viele Menschen“ oder aus „14%“ „einige“ oder „wenige“

Sätze

- **S1. Benutzen Sie kurze Sätze**:
  Jeder Satz solle nur eine Aussage enthalten. Als Beispiel wird aus dem folgenden Satz

  > „Ich habe meinem guten Freund Leo
  > ein Buch über die Geschichte
  > von Berlin geliehen.“

  drei Sätze mit jeweils nur einer Aussage

  > „Leo ist ein guter Freund von mir.
  > Ich habe ihm ein Buch geliehen.
  > Das Buch ist über die Geschichte von Berlin.“

- **S2. Benutzen Sie einen einfachen Satzbau**:
  Wörter sollen der Subjekt-Prädikat-Objekt-Reihenfolge folgen, sodass beispielsweise zuerst der Akteur (Subjekt) und dann die Handlung (Prädikat) benannt wird. Statt „Die Rechnung bezahlt Frau Weber.“, solle „Frau Weber bezahlt die Rechnung.“ genutzt werden.

- **S3. Sie dürfen verkürzte Sätze benutzen**:
  Sätzen müssen nicht vollständig sein und dürfen verkürzt werden. Zudem dürfen Sätze mit „Oder“, „Und“ oder „Aber“ beginnen. So kann aus dem Satz

  > „Wollen Sie nach Berlin
  > oder nach Hamburg fahren?“

  folgende zwei Sätze entstehen

  > „Wollen Sie nach Berlin fahren?
  > Oder nach Hamburg?“

Insgesamt soll, wie demonstriert, nach jedem Satz eine neue Zeile begonnen werden, was über die Regel „G5. Schreiben Sie jeden neuen Satz in eine neue Zeile“ [111, S. 50] vorgeschlagen wird.

2.1.4 Evaluation von Leichter Sprache

Zur Beurteilung von Texten in Leichter Sprache hinsichtlich ihrer Verständlichkeit und ihres Vereinfachungsgrades können qualitative und quantitative Methoden verwendet werden.

Qualitative Methoden basieren auf verbalen Rückmeldungen von Studienteilnehmern, wie durch Interviews mit Fragen zum Text oder Wiedergabeverfahren. Durch Gespräche und verbale Reflexionen sollen Erkenntnisse aus dem Leseverständnis gewonnen werden [14, S. 213-241]:
Durch gezielte Fragen zum Text können sowohl Faktenwissen als auch ein tieferes Textverständnis direkt und strukturiert geprüft werden. Wiedergabeverfahren hingegen zielen auf eine indirekte und freiere Reproduktion des Textverständnisses ab, wodurch die individuelle Repräsentation des Textes im Gedächtnis und die Gewichtung der Informationen erkennbar werden.

Quantitative Methoden basieren hingegen auf experimentellen Verfahren, wie lexikalische Entscheidungen, Self-paced reading oder Blickbewegungsmessung, und vor allem Lesbarkeitsindizes, um die Lesbarkeit von Texten messbar und objektiv zu erfassen. Diese Methoden zielen darauf ab, durch messbare Daten und experimentelles Vorgehen das Leseverhalten und die Textverständlichkeit quantitativ zu analysieren [14, S. 188-203].
Experimentelle Verfahren beinhalten die Manipulation von Variablen (z.B. Wortlänge)

und die Messung von Reaktionen (z.B. Reaktionszeit, Blickbewegungen), um Rückschlüsse auf kognitive Prozesse beim Lesen zu ziehen. Bei lexikalischen Entscheidungen wird die Reaktionszeit gemessen, um zu beurteilen, ob eine Buchstabenkette ein echtes Wort ist. Self-paced reading erfasst die Lesezeit für einzelne Segmente (z.B. Wörter), indem die Teilnehmer selbst den Lesefortschritt per Tastendruck steuern. Die Blickbewegungsmessung erfasst Augenbewegungen beim Lesen, wodurch Rückschlüsse auf Verarbeitungsaufwand und Leseverständnis gezogen werden können. Lesbarkeitsindizes sind Metriken, die die Lesbarkeit bzw. Einfachheit eines Textes auf Basis von Wort- und Satzlängen bestimmen. Erste Lesbarkeitstests sind bereits seit 1923 bekannt, jedoch erwies sich das Flesch-Reading-Ease Verfahren [51] als „besonders einflussreich“ [19, Kapitel 2.1.1].
Flesch-Reading-Ease quantifiziert die Lesbarkeit eines Textes, wobei höhere erzielte Werte auf eine höhere Lesbarkeit hindeuten. Dies basiert auf der durchschnittlichen Satzlänge und der Silbenanzahl pro Wort.
Weitere Lesbarkeitsindizes für deutschsprachige Texte stellen der Hohenheimer Verständlichkeitsindex und die Wiener Sachtextformel dar [19, Kapitel 2.1.1]. Während der Hohenheimer Verständlichkeitsindex vor allem fachsprachliche Texte evaluiert, dient die Wiener Sachtextformel der Einschätzung der Verständlichkeit von Sachtexten. Sie berücksichtigt u.a. die durchschnittliche Satzlänge und die Anzahl von Wörtern mit mehr als drei Silben und existiert in verschiedenen Varianten.

2.2 Monolinguale Maschinenübersetzung

2.2.1 Grundlagen von Maschinenübersetzung

Maschinenübersetzung bezeichnet automatisierte Systeme, die mithilfe von Computern einen Text von einer Quellsprache auf eine Zielsprache übersetzen.
Die grundlegende Idee der Maschinenübersetzung stammt aus dem Weaver Memorandum von 1949. In dieser Veröffentlichung griff Warren Weaver erstmals die Möglichkeiten von Übersetzungen mithilfe von Computern auf und erörterte sie [166]. Seitdem entwickelten sich verschiedenste Systeme, die unterschiedliche Methoden zur Übersetzung nutzen.

Regelbasierte Methoden

Systeme mit regelbasierten Methoden nutzen explizite linguistische Regeln (Grammatik, Morphologie, Syntax oder Semantik), um Texte zu übersetzen und sind in den meisten

Fällen domänenspezifisch. Die Regeln werden dabei von Fachexperten oder Linguisten unter hohem Aufwand manuell gepflegt. Eines der ersten regelbasierten Systeme entstammt aus dem Georgetown-IBM Experiment im Jahre 1954. Als einer der ersten Maschinenübersetzer, welcher in der Öffentlichkeit großes Interesse erregte, war es darauf konzipiert mit dem IBM 701 Computer russische Sätze ins Englische zu übersetzen [73]. Mit 250 Wörtern und sechs Regeln zur Grammatik wurden 60 ausgewählte russische Sätze ins Englische übersetzt, was zu „Optimismus hinsichtlich einer raschen Entwicklung in der nahen Zukunft“ [73] von Maschinenübersetzern führte. Ein weiteres nennenswertes System ist Systran, der älteste vollumfängliche Maschinenübersetzer, der anfangs ebenfalls regelbasiert war [156].
Es wird insgesamt zwischen drei Arten von regelbasierten Systemen unterschieden [121, Kapitel 3], den Direkten-, Transfer- und Interlingualen Systemen, die nach Bernard Vauquois [159] als Dreieck repräsentiert werden können, siehe Abbildung 2.1.
Direkte Systeme übersetzen direkt von der Eingabe zur Ausgabe mithilfe einfacher Regeln und Wörterbücher. Transfersysteme auf der anderen Seite verwenden syntaktische Analysen, um die Eingabesprache in eine Zwischenrepräsentation zu überführen, die dann in die Zielsprache übersetzt wird. Schließlich nutzen interlinguale Systeme eine abstrakte, sprachunabhängige Bedeutung („Interlingua“) als Zwischenrepräsentation, bevor in die Zielsprache übersetzt wird.

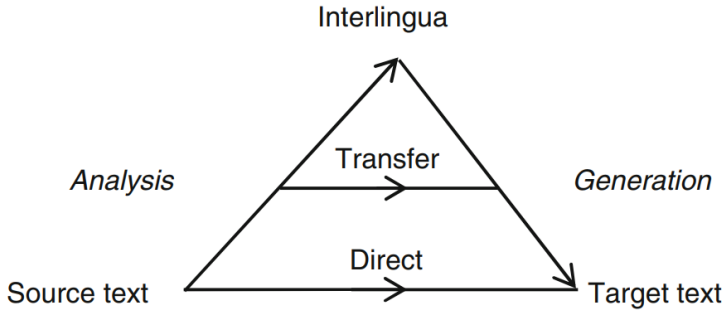

Abbildung 2.1: Simplifizierte Darstellung des Vauquois Dreiecks aus [105] nach [159]

Der generelle Ansatz besteht dabei aus einer Analyse-, einer Transfer- und einer Generierungsphase, bei dem zunächst die Ausgangssprache analysiert, dann entweder direkt, durch Transfer- oder Interlingua-Methoden transferiert und schließlich die Zielsprache auf Basis dessen generiert wird.

Beispielbasierte Methoden

Beispielbasierte Systeme nutzen zur Übersetzung in eine Zielsprache vorübersetzte Beispiele, die in einem bilingualen Korpus vorliegen, mithilfe von Analogie. Im Gegensatz zu regelbasierten Systemen sind keine expliziten linguistischen Regeln von Nöten. Eines der ersten Konzepte eines beispielbasierten Maschinenübersetzersystems stammt von Nagao [107], der die ersten Ansätze eine Übersetzung anhand von Analogiebeispielen zwischen dem Japanischen und dem Englischen skizzierte. Die grundlegende Idee der Analogieübersetzung beschreibt Nagao folgendermaßen:

> „(1) Man does not translate a simple sentence by doing deep linguistic analysis, rather,
> (2) Man does the translation, first, by properly decomposing an input sentence into certain fragmental phrases (very often, into case frame units), then, by translating these fragmental phrases into other language phrases, and finally by properly composing these fragmental translations into one long sentence. The translation of each fragmental phrase will be done by the analogy translation principle with proper examples as its reference [...]." [107]

Daraus lassen sich drei Hauptkomponenten beispielbasierter Methoden ableiten: der Abgleich von Textfragmenten mit einer Datenbank realer Beispiele, die Identifikation von entsprechenden Übersetzungsfragmente und das abschließende Zusammenfügen des Zieltexts [140].

Nach Somers [140] lässt sich anhand des Vauquois-Dreiecks die Aufgaben von beispielbasierten Methoden darauf „überlagern", siehe Abbildung 2.2.

Der Ansatz für beispielbasierte Methoden beinhaltet zunächst die Substitution des Analyseschritts des Ausgangstexts durch den Abgleich innerhalb einer Beispieldatenbank („Matching"). Die ausgewählten Beispiele werden im Zieltext adaptiert („Alignment") und zu einem vollständigen Text zusammengefügt, ähnlich der Generierung im Ursprungsansatz („Recombination"). Der exakte Abgleich („Exact Match") minimiere zwar die Analyse, könne aber aufgrund seiner idealen Repräsentation oben im Dreieck gesehen werden [140].

Ein dazugehöriges Konzept ist das des TM, auch Übersetzungsspeicher genannt, welches im Wesentlichen die genannte Beispieldatenbank darstellen kann und die bereits übersetzte Textsegmente zusammen mit ihren Originalversionen speichert. Obwohl der Ursprung

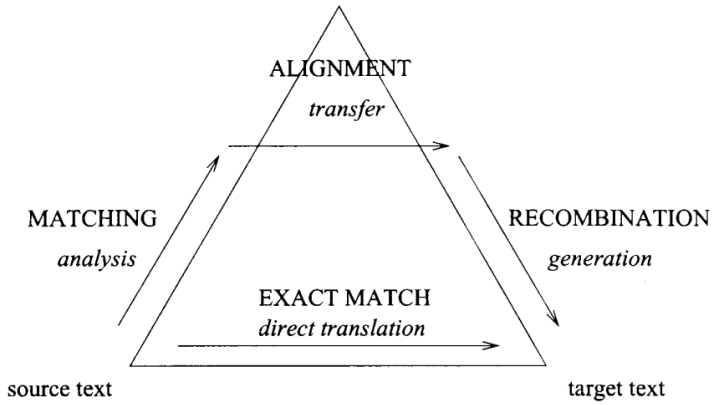

Abbildung 2.2: Vauquois Dreieck adaptiert für beispielbasierte Methoden aus [140] nach [159]

der Grundidee des TM nicht eindeutig bestimmbar ist [140], lassen sich einflussreiche Erstimplementationen der Kernansätze 1990 datieren [128]. Unter diesen Ansätzen gehört der Begriff des Translation Unit (TU), welcher als Basiseinheit Ausgangs- und Zieltext als „Bi-Text“ verknüpft und abspeichert, wobei der Abruf einer Ausgangseinheit die dazugehörige Zieleinheit mit sich bringt [63]. Diese Basiseinheiten können von einzelne Wörter bis hin zu ganzen Sätzen reichen.
Abruf und Abgleich des TU sind ein weiterer Aspekt beim TM. Um Übersetzungsergebnisse zu finden, müssen die TU aus dem TM abgerufen und mit unterschiedlichen Algorithmen abgeglichen werden. Dabei lässt sich der Prozess in drei Kategorien einteilen [35]: exakte, „fuzzy“- oder „full“- [139] sowie keine Übereinstimmung beim Abgleich. Die zugrundeliegenden Abgleichsalgorithmen basierten dabei größtenteils auf einfachen Ähnlichkeitsberechnungen von Zeichenketten, die als „Edit-Distance“ bezeichnet werden [139]. Sie gibt die minimale Anzahl von Ersetzungen, Einfügungen und Löschungen an, die erforderlich sind, um eine Zeichenkette in eine andere umzuwandeln. Andere fortschrittlichere Algorithmen bezogen aber auch schon neuronale Netzwerke mit ein. Die meisten Systeme auf Basis von TMs definieren nach Christiansen und Schjoldager eine exakte Übereinstimmung als 100 %, eine Fuzzy-Übereinstimmung zwischen 70 % und 99 % sowie keine Übereinstimmung bei unter 70 % [35].

Grundsätzlich lassen sich TMs und beispielbasierten Methoden von Maschinenübersetzungen dahingehend unterscheiden, dass ein TM auch und vor allem in Verbindung mit Menschen eingesetzt wird, die man als Computer-assisted Translation (CAT)-Methoden bezeichnen kann [35]. Während TMs als interaktive Tools für menschliche Übersetzer

angesehen werden können, seien Maschinenübersetzer, die auf Beispielen basieren, grundsätzlich automatisiert, wobei unterschiedliche Auffassungen herrschen [140].

Statistische Methoden

Neben den beispielbasierten Methoden entstanden parallel auch statistische Methoden, die auf Basis von Modellen die Wahrscheinlichkeit des Zieltextes auf Grundlagen des Ausgangstextes bestimmen. Parallele Korpora, als Sammlung von Texten mit ihren Übersetzungspaaren [82, S. 54], bilden dabei die Grundlage, mit der die statistischen Modelle trainiert werden. Erste experimentelle Versuche begannen im Jahre 1990 [21], bei der Sätze aus dem Französischen ins Englische durch Ermittlung einer Wahrscheinlichkeit übersetzt wurden. Kernkomponenten, siehe Abbildung 2.3, waren ein Sprachmodell zur Zuweisung von Wahrscheinlichkeiten an Ausgangssätzen, ein Übersetzungsmodell zur Berechnung der Wahrscheinlichkeit des Zielsatzes auf Basis des Ausgangssatzes und ein Decoder für die eigentliche Übersetzungsarbeit, bei der der Satz, für den die Wahrscheinlichkeit am Größten ist, ausgewählt wurde.

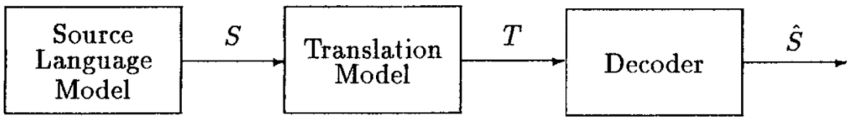

Abbildung 2.3: Statistisches Maschinenübersetzungssystem adaptiert nach Brown et al. [21]

Auf Grundlage dieser Arbeit entwickelten Brown et. al [21] fünf statistische Modelle mit steigender Komplexität, um den Übersetzungsprozess zu modellieren. Ein zentrales Konzept ist das Wort-für-Wort-Alignment zwischen englischen und französischen Sätzen zur Modellierung der Beziehungen zwischen den Wörtern [22]. Diese Wort-für-Wort Modelle entwickelten sich im weiteren Verlauf zu phrasenbasierten Modellen, die Segmente aus Wörtern betrachteten, die Übersetzungsqualität verbesserten und weitreichend adaptiert wurden [162]. Um Modelle dieser Art trainieren zu können, stellen der Alignmentprozess sowie der parallele Korpus wesentliche Aspekte dar.
Neben dem Alignment von Wörtern in den Anfangsphasen der statistischen Methoden, existiert auch das Alignment von Sätzen als essentielle Basis der parallelen Korpora.
Satzalignment bzw. Sentencealignment stellt den Prozess dar, Übersetzungspaare aus Sätzen anhand von Dokumenten zu ermitteln, siehe Abbildung 2.4. Dieser Prozess ist

nicht immer trivial, da „Text selten Wort für Wort und nicht immer Satz für Satz übersetzt wird“, sondern Sätze zerlegt oder verschmolzen werden und verschiedene Sprachen unterschiedliche Satzkonzepte besitzen [82, S. 55].
Aus diesem Grund wurde diese Thematik schon in den Anfangsphasen der statistischen Methoden untersucht, u.a. durch Brown et al. [20] und Gale und Church [55]. Diese Methoden nutzen vor allem die Satzlänge als Anhaltspunkt für das Alignment. Andere Methoden verwenden „Ausrichtungsketten, modellieren Auslassungen oder unterscheiden grobe Segmentierung von Text und detailliertem Sentencealignment“. Auch können lexikalische Informationen, Inhaltswörter, Zahlen und n-Gramme berücksichtigt werden [82, S. 60]. Modernere Ansätze nutzen sogar Maschinenübersetzer, wie bei Sennrich und Volk [136], oder vergleichen Sätze auf Basis von Vektoren, die nach Größe der Übereinstimmung verknüpft werden, wie bei Thompson und Koehn [153].

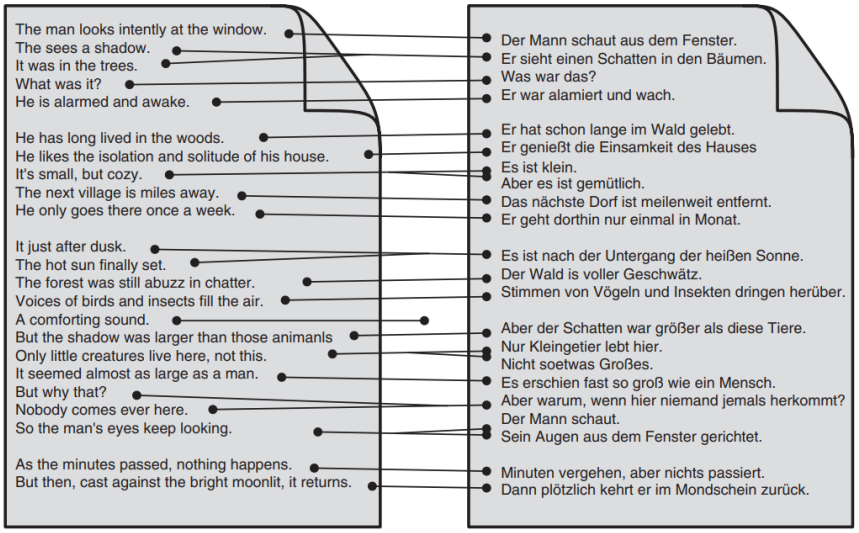

Abbildung 2.4: Sentencealignment anhand eines Dokuments und seiner Übersetzung [82, S. 56]

Neben den Übersetzungsmodellen sind vor allem Sprachmodelle („Language Models“) essentieller Bestandteil von allen Übersetzungssystemen mit statistischer Methoden. Language Models bewerten den Lesefluss von generierten Texten und wirken sich auf die Wortwahl, die syntaktische Struktur und andere Aspekte der Textgenerierung aus. Sie weisen Sätzen Wahrscheinlichkeiten zu, um diese in Texten vorhersagen zu können und nutzen n-Gramme und Markov-Annahmen, um diese zu berechnen [82, S. 9-10].
n-Gramme, als Konzept des NLP, können Sequenzen von n Wörtern bezeichnen oder auch ein Wahrscheinlichkeitsmodell, welches die Wahrscheinlichkeit eines Wortes anhand

der n-1 vorherigen Wörter bestimmt. Mithilfe von Markov-Ketten wird die Annahme getroffen, dass die Wahrscheinlichkeit eines Wortes nur anhand des vorherigen Wortes ermittelt werden kann. Diese Annahme wird auch als Markov-Annahme bezeichnet. Da die ersten Language Models n-Gramm basiert sind, nennt man sie auch „N-gram Language Models“ [77, S. 33-35].

Neuronale Methoden

Auf Basis der statistischen Methoden entstanden neuronale Methoden als Fortführung statistischer Verfahren zur Übersetzung. Im Gegensatz zu rein statistischen Methoden werden bei neuronalen Methoden neuronale Netzwerke genutzt, um die Übersetzung zwischen Ausgangs- und Zieltext zu erlernen. Da statistische Maschinenübersetzer, trotz ihrer hohen Verbreitung, auf eine komplizierte Integration vieler manuell entworfener statistischer Modelle beruhen, können umfangreiche, parallele Korpora nicht optimal genutzt werden, was zu einer unzureichenden Übersetzungsqualität führt [162].
Durch die Entwicklung der Encoder-Decoder-Architektur 2014 [34, 149] wurden die Limitationen durch die Nutzung jeweils eines neuronalen Netzwerks als Encoder und Decoder gemindert, siehe Abbildung 2.5.

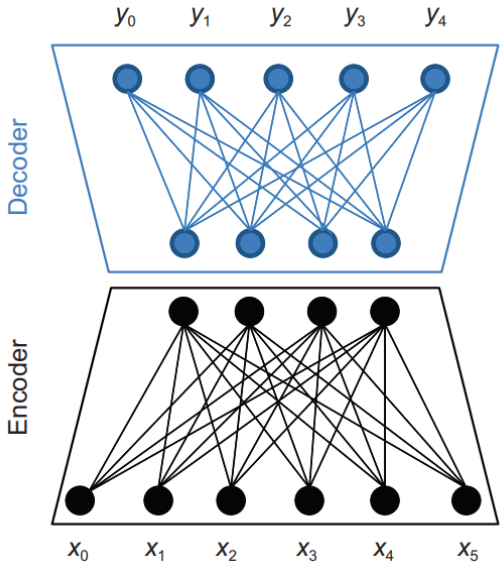

Abbildung 2.5: Encoder-Decoder Architektur eines neuronalen Maschinenübersetzers [174]

Der Encoder in der Architektur überführt dabei den Ausgangssatz in eine Vektorrepräsentation aus welchem der Decoder die Übersetzung Wort für Wort auf Grundlage des Verständnisses des Ausgangssatzes generiert. Dies hat den Vorteil, dass, im Gegensatz zu regelbasierten oder statistischen Methoden, keine manuell gepflegten Regeln oder Funktionen notwendig sind [162].
Mit der weiteren Verbesserung der Architektur durch Nutzung einer „automatischen (soften) Suche für Bestandteile des Ausgangssatzes, die relevant für die Vorhersage eines Zielwortes [...]“ sind [9], der „Attention“-Mechanismus, wurde der Weg für die heutigen LLMs in Form der Transformer geebnet [158]. Diese werden unter Abschnitt 2.3 näher beschrieben.

2.2.2 Grundlagen von automatischer Textvereinfachung

Eine spezielle Form einer Maschinenübersetzung stellt die automatisierte Textvereinfachung dar. Sie setzt sich zum Ziel, die lexikalische und strukturelle Komplexität von Texten automatisiert in eine einfachere Version zu überführen, während gleichzeitig die Kernbedeutung des Textes erhalten werden soll. Basierend auf der Klassifizierung von Al-Thanyyan [151] lassen sich automatisierte Textvereinfachungsverfahren in vier Klassen einteilen: lexikalische, syntaktische, monolinguale Maschinenübersetzung oder hybride Techniken.
Nach Al-Thanyyan konzentriert sich die lexikalische Vereinfachung auf die Identifikation komplexer Wörter und deren Ersetzung durch einfachere Synonyme und operiert zumeist auf Wortebene, wodurch übergeordnete Textkohäsion und -kohärenz nicht immer ausreichend berücksichtigt werde. Ihr Hauptziel ist die Verbesserung der Textzugänglichkeit für Leser mit unterschiedlichen Literalitätsniveaus.
Im Gegensatz dazu ziele die syntaktische Vereinfachung darauf ab, komplexe syntaktische Strukturen, wie Koordination, Subordination, Relativsätze und Passivkonstruktionen, zu vereinfachen, ohne die ursprüngliche Bedeutung zu verändern. Dies wird durch die Modifikation der Syntax erreicht. Dadurch wird der Text verständlicher, ohne dass Informationen verloren gehen.
Des Weiteren nutzen Systeme zur monolingualen Maschinenübersetzung statistische oder neuronale Methoden, um komplexe Sätze innerhalb derselben Sprache in einfachere Formen zu überführen. Diese stützen sich auf Fortschritte im NLP und dem Deep Learning, um die Lesbarkeit zu erhöhen, während die ursprüngliche Bedeutung erhalten bleibt.
Schließlich kombinieren hybride Techniken datengesteuerte lexikalischen mit regelbasier-

ten syntaktischen Vereinfachungsverfahren, um die Stärken beider Ansätze zu nutzen und so eine effektivere Textvereinfachung zu erreichen. Monolinguale Maschinenübersetzungssysteme zur Textvereinfachung nutzen dabei in den jüngsten Studien entweder statistische oder neuronale Methoden zur Übersetzung von komplexen in simplere Texte [151].

Die erste statistische Methode zur Textvereinfachung wurde von Specia [142] vorgestellt, bei der durch einen portugiesischen parallelen Korpus auf Satzebene ein statistisches Modell zur Übersetzung trainiert wurde. Das Modell lieferte ein vielversprechendes Ergebnis, da die Gesamtqualität der Textvereinfachung erhalten blieb und insbesondere lexikalische Vereinfachungen gut umgesetzt wurden. Einen weiteren Ansatz verfolgte Zhu et al. [182] mit einem baumbasierten statistischen Modell, das zur damaligen Zeit erstmalig das Aufteilen von Sätzen, das Weglassen von Satzteilen, das Umordnen von Satzstrukturen und den Austausch von Wörtern oder Phrasen abdeckte. Dieser Ansatz wählte einen parallelen englischen Wikipedia Korpus auf Satzebene und erreichte bessere Lesbarkeitswerte als damalige SOTA-Systeme. Auch der Ansatz von Coster und Kauchak [36] nutzte einen englischen parallelen Wikipedia Korpus und erweiterte durch ihr trainiertes Übersetzungsmodell einen phrasenbasierten Maschinenübersetzer, welcher die Löschung von Phrasen beinhaltete. Die Ergebnisse des Ansatzes waren im Vergleich zu anderen Textkompressionstechniken und dem Modell ohne die Löschung von Phrasen signifikant besser. Xu et al. [169] stellten fest, dass vor allem der parallele Wikipedia Korpus in vielen Fällen für diese Ansätze genutzt worden ist, jedoch „Unzulänglichkeiten“ aufwies. Sie stellten nicht nur einen neuen parallelen Korpus, sondern auch einen neuen Ansatz vor [170]. Dieser ist nicht nur auf manuell erstellte Korpora angewiesen, da diese limitiert sind aufgrund ihrer Qualität, Quantität und des Aufwands zur Erstellung. Bei diesem Ansatz wurden stattdessen große, zweisprachige Korpora genutzt, aus denen automatisiert Regeln für die Generierung von Paraphrasen erstellt wurden. Zusätzlich wurden Vereinfachungsmerkmale sowie unterschiedliche Referenzvereinfachungen zur Qualitätssicherung inkludiert. Auch dieser Ansatz zeigte signifikante Verbesserungen bei der Textvereinfachung.

Durch die Entwicklung der neuronalen Methoden, ergaben sich auch in diesem Bereich erste Ansätze zur Textvereinfachung. So untersuchten Ende 2016 Nisioi et al. [116] und Wang et al. [164] mit den ersten experimentellen Ansätze die Möglichkeit von trainierten Encoder-Decoder Modellen zur Textvereinfachung. In beiden Ansätzen zeigte sich, dass neuronale Methoden zur Textvereinfachung die vorherigen SOTA-Modelle auf Basis statistischer Methoden übertrafen. In beiden Versuchen wurde ebenso geschlussfolgert, dass diese Modelle die Regeln der Textvereinfachung aus den Daten erlernen und anwenden

könnten. Zhang und Lapata [178] kombinierten den Ansatz von Encoder-Decoder Modellen mit Deep Reinforcment Learning, bei der das Encoder-Decoder Modell als Agent betrachtet wird und bei der Produktion eines vereinfachten Satzes über ein Reinforcement Algorithmus das Modell je nach Belohnungswert aktualisiert. Auf Basis dreier Datensätze wurde die bessere Leistung dieser Kombination gegenüber anderen Textvereinfachungssystemen aufgezeigt. Viele weitere Ansätze mit neuronalen Methoden wurden in dieser Zeit vorgestellt, mehrheitlich für die Textvereinfachung von englischsprachigen Texten [46].
Auch Systeme auf Basis der Transformer Modelle wurden umgesetzt, welche unter Abschnitt 2.3 im weiteren Verlauf näher erörtert werden.

2.2.3 Evaluation von monolingualer Maschinenübersetzung

Die Evaluation von Texten aus monolingualen Maschinenübersetzungen ist im Vergleich zu anderen NLP-Aufgaben, wie Information Retrieval (IR) oder Frage-Antwort-Systemen, komplexer. Während die genannten Aufgaben eindeutige, faktische Antworten anstreben, ist die Textvereinfachung subjektiv und schwer standardisierbar, da sie stark vom individuellen Wissen und der Auffassungsgabe des Lesers abhängt. Dazu haben verschiedene Faktoren einen gewissen Einfluss: die Rolle des Endnutzers, die Referenzdaten, die Domänen der Ausgangsdokumente und die Evaluationsmetriken selbst [58].
Die Evaluation lässt sich grundsätzlich in zwei Kategorien einordnen: Die menschliche (manuelle) Evaluation beurteilt Texte anhand ihrer Semantik, Grammatikalität und ihrer Einfachheit [58, 151]. Im Bereich der Semantik wird beurteilt, ob und wie gut der Text nach der Übersetzung seine Bedeutung beibehält. Die Grammatikalität betrachtet die grammatikalische Korrektheit und die allgemeine Verständlichkeit. Mit der Einfachheit wird schließlich überprüft, ob der Zieltext einfacher zu lesen ist, als der Ausgangstext. Die Messung dieser Metriken wird durch menschliche Experten über Likert-Skalen durchgeführt, innerhalb einer Spanne von 0-5 oder auch zwischen -2 bis +2 [3, 151]. Diese Art der Messung ist subjektiv bzw. inkonsistent, da bei den beurteilenden Personen linguistisches Wissen vorausgesetzt wird und verschiedene Personen unterschiedlich bewerten würden [3, 58, 151].

Eine maschinelle und automatisierte Evaluation lässt sich einerseits in referenzbasierte und andererseits in nicht-referenzbasierte Verfahren unterteilen [98]. Referenzbasierte Metriken nutzen den Ausgangstext als Referenz, um den Zieltext auf Basis dessen zu evaluieren [58]. Unter den bekanntesten und meistgenutzten, die n-Gramm basiert sind,

gehören die Metriken Bilingual Evaluation Understudy (BLEU) [118] und System output Against References and Input (SARI) [170].
BLEU bewertet die Übereinstimmung von n-Grammen zwischen dem Zieltext und den Referenztexten. BLEU konzentriert sich hauptsächlich auf die Präzision, d.h. wie viele der n-Gramme im Zieltext auch in den Referenztexten vorkommen. SARI bewertet die Qualität eines vereinfachten Textes, indem die Metrik den Grad der Hinzufügung, Löschung und Beibehaltung von Informationen im Vergleich zum Originaltext analysiert. Sie misst somit, wie gut ein Vereinfachungssystem den Sinn und die wesentlichen Inhalte beibehält.

Es existieren jedoch auch Metriken, die nicht n-Gramm basiert sind, darunter z.B. BERTScore [177], welches die kontextbezogenen Einbettungen des Bidirectional Encoder Representations from Transformers (BERT)-Modells [40] nutzt, um die semantische Ähnlichkeit zwischen dem Zieltext und dem Referenztext zu beurteilen. Dies ermöglicht eine differenziertere Bewertung der Textqualität im Vergleich zu n-Gramm Metriken.
Insgesamt korrelieren die Metriken BLEU und BERTScore mit der Grammatikalität sowie Semantik und weniger mit Einfachheit, während die SARI Metrik mehr mit Einfachheit und Grammatikalität korreliert [58, 98, 168, 170].

Nicht-referenzbasierte Metriken wiederum nutzen keine Referenzen auf Basis des Ausgangstextes, sondern den Zieltext selbst, und können nach Martin et al. vor allem zur Bewertung der Einfachheit eines Textes eingesetzt werden [98]. Sie stellten unter Vorbehalt ebenfalls fest, dass die Satzlänge ein entscheidender Faktor ist und die Anzahl der Zeichen, Silben und Wörter ein valides Maß für die Einfachheit darstellt. Zu diesen nicht-referenzbasierten Metriken gehören u.a. Lesbarkeitsindizes wie Flesch-Reading-Ease, Flesch-Kincaid-Grade-Level oder auch FOG bzw. SMOG [151]. FOG kombiniert zur Messung die durchschnittliche Länge der Sätze mit der durchschnittlichen Anzahl komplexer Wörter pro 100 Wörter Text, während SMOG ähnlich zu FOG ist, jedoch nur mit der durchschnittlichen Anzahl mehrsilbiger Wörter in Textsegmenten von 30 Sätzen Textabschnitten kalkuliert.

2.3 Large Language Models

2.3.1 Grundlagen von LLMs

Wie bereits in Abschnitt 2.2.1 beschrieben, basieren die heutigen LLMs zu großen Teilen auf der in 2017 vorgestellten Transformer-Architektur, die eine gestapelte Encoder-Decoder-Architektur mit Nutzung des Attention-Mechanismus kombiniert [158].

Die neuronalen Netzwerke der Encoder und Decoder in Abschnitt 2.2.1, die genutzt wurden zur Maschinenübersetzung, waren zumeist Rekurrente Neuronale Netzwerke (RNN) und ihre Weiterentwicklung Long Short-Term Memory (LSTM)-Netzwerke [67], die im Vergleich zu den Transformer-Netzwerken einige Limitierungen aufweisen. RNN sind Netzwerke, die Daten iterativ sequenziell verarbeiten, indem Informationen in vorherigen Schritten in einem versteckten („hidden“) Zustand zwischengespeichert werden. Dadurch können zeitliche Abhängigkeiten, wie bei Wörtern in natürlicher Sprache, modelliert werden. Jedoch leiden diese Netzwerke unter dem Problem, dass langfristige Abhängigkeiten aufgrund des „vanishing and exploding gradient“-Problems [119] nicht erlernt werden können. LSTM-Netzwerke mindern diese Problematik durch die Nutzung von Toren („Gates“), die steuern, welche Informationen in Gedächtniszellen („Memory cells“) gespeichert werden [67]. Dadurch können sie längerfristige Abhängigkeiten in Sequenzen besser erfassen, sind jedoch weiterhin sequenziell bzw. rekurrent in ihrer Verarbeitung.

Die Transformer-Architektur löst einige Limitationen der RNN und der LSTM-Netzwerke, indem sie Informationen nicht rekurrent verarbeitet, sondern in einer parallelisierten Form. Dadurch können unabhängig von der Sequenzlänge längerfristige Abhängigkeiten, vor allem durch einen „Self-Attention“-Mechanismus, erfasst werden. Zudem ist das Training des Netzwerks hochgradig parallelisierbar und signifikant schneller [158]. Die von Vaswani et al. vorgestellte Architektur, siehe Abbildung 2.6, verfügt jeweils über sechs identische Encoder sowie Decoder, welche ebenfalls mehrere Self-Attention-Mechanismen und „Feed Forward“-Netzwerke beinhalten. Dadurch war es möglich, neue SOTA Übersetzungsqualität von Texten aus Englisch zu Deutsch und Französisch zu Englisch zu erreichen.

Die grundlegenden Komponenten der Architektur und ihrer Funktionsweise werden im Folgenden beschrieben [158] (siehe auch [77, S. 184-201]:

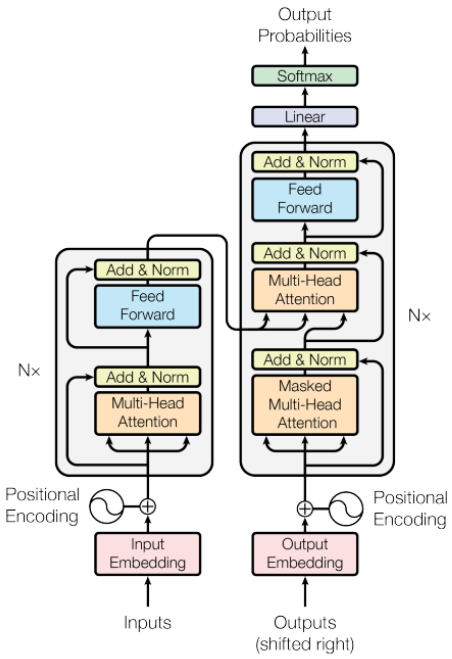

Abbildung 2.6: Transformer-Architektur nach Vaswani et al. [158]

Tokenization

Bevor NLP-Aufgaben angegangen werden können, müssen Texte zunächst normalisiert werden. Einer dieser Wege ist der Tokenizationprozess, bei dem eine Textsegmentierung stattfindet, um kleinste prozessierbare Texteinheiten zu erhalten. Neben der Möglichkeit regelbasiert Texteinheiten zu erhalten („Top-down tokenization“), nutzten Vaswani et al. eine auf statistische Methoden beruhende Variante („Bottom-up tokenization“), die auch in den meisten heutigen LLMs Anwendung findet. Dabei werden Wörter in Teilwort-Token („subword tokens“) aufgeteilt, welche Wörter, Teilwörter oder nur einzelne Zeichen darstellen, wodurch auch selten benutzte Wörter durch Zusammenlegung von Teilwörtern repräsentiert werden können [77, S.18-23].
Der von Vaswani et al. verwendete Algorithmus ist der Byte-Pair-Encoding (BPE)-Algorithmus [137], welcher auf dem BPE-Kompressionsalgorithmus von Gage basiert [54] und für Wortsegmentierungen adaptiert wurde. Der Algorithmus lernt dabei die Zusammenführung häufiger Symbolpaare in einem Vokabular, um neue n-Gramme zu erstellen. Er beginnt mit einzelnen Zeichen und führt iterativ das häufigste Paar zu einem neuen Symbol zusammen, wodurch die Vokabulargröße erhöht wird. Dies wird für eine festge-

legte Anzahl k von Zusammenführungsoperationen fortgesetzt. Die Zusammenführungen sind auf Wortgrenzen beschränkt und erzeugen Teilwort-Einheiten, die eine Verallgemeinerung auf unbekannte Wörter ermöglichen.

Embedding

Nach der Textsegmentation durch den Tokenisierungsprozess werden Wörter mittels Embeddings in eine Vektorrepräsentation kodiert. Eine der bekanntesten Methoden zur Umsetzung von Wörtern in Vektoren ist *word2vec* [101], welches im engeren Sinne eine Software aus zwei Bestandteilen, Continuous Bag of Words (CBOW)- und Skip-gram-Architekturen, darstellt, um Word Embeddings zu ermöglichen. Während die CBOW-Architektur die Vorhersage eines Wortes aus dem Kontext trifft, arbeitet die Skip-gram-Architektur mit jedem Wort, um auf dessen Basis Vorhersagen über die umliegenden Wörter in einer bestimmten Wortentfernung zu treffen, siehe Abbildung 2.7. Beide trainieren ein neuronales Netzwerk, um diese Vorhersagen treffen zu können, welches die Basis für die Word Embeddings bietet.

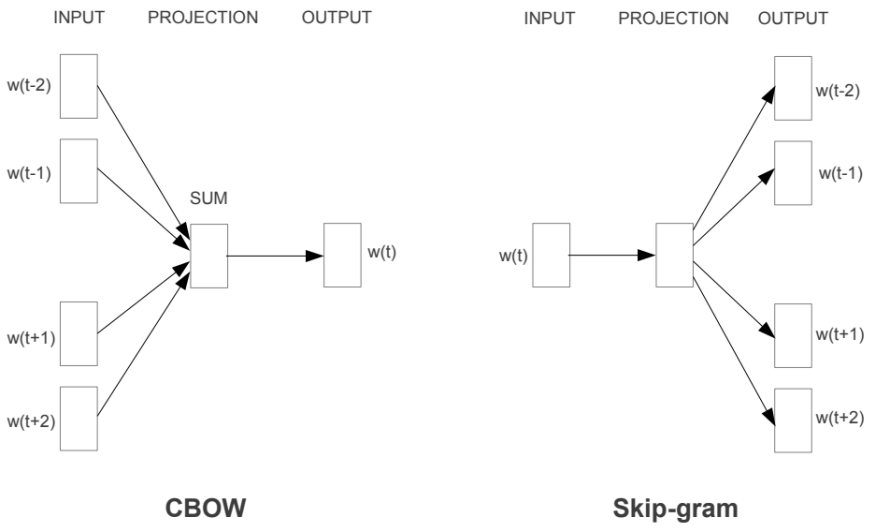

Abbildung 2.7: CBOW- und Skip-gram-Architektur [101]

Positional Encoding

Nachdem Wörter in Vektoren enkodiert wurden, werden die einzelnen Positionen der Wörter in einer Sequenz ebenfalls enkodiert. Transformer-Architekturen unterscheiden sich von den klassischen Encoder-Decoder Varianten mit RNN und LSTM-Netzwerken durch eine fehlende Rekurrenz. Dadurch sind sie aufgrund ihrer Struktur nicht dazu in der Lage sequentielle Informationen in ihrer Reihenfolge zu bearbeiten. Stattdessen werden dazu Positional Encoding-Mechanismen genutzt [103]. Positional Encoding erlaubt es Transformern ohne Rekurrenz die „relative oder absolute Position von Token in einer Sequenz“ [158] zu erhalten, um die Reihenfolge der Sequenz nutzen zu können.

Multi-Head Attention

Einer der wichtigsten Komponenten in Transformer-Architekturen ist der (Self-)Attention-Mechanismus, welcher die Wortbeziehungen innerhalb der Eingabesequenz modelliert.
Der Self-Attention-Mechanismus, bei Vaswani et al. als „Scaled Dot-Product Attention“ bekannt, modelliert Beziehungen zwischen Elementen einer Sequenz in einer parallelisierten Form. Genauer wird eine gewichtete Summe von Vektorrepräsentationen der Eingabesequenz berechnet. Diese Modellierungsfähigkeit globaler Abhängigkeiten innerhalb der Eingabe eliminiert die Notwendigkeit für rekursive Schichten, die traditionell zur Erfassung von Langzeitabhängigkeiten verwendet werden. Außerdem wird dadurch eine Parallelisierung bei der Verarbeitung langer Sequenzen ermöglicht.
In der von Vaswani et al. vorgestellten Architektur wurden acht Attention-Mechanismen hintereinander gestapelt, was als Multi-Head Attention bezeichnet wird. Diese ermöglichen es, komplexere Beziehungen und Abhängigkeiten in Sequenzen zu erfassen.

Während der Encoder in dieser Form modelliert wurde, wurde in der Originalarchitektur der Decoder mit einem maskierten Multi-Head Attention-Mechanismus erweitert, welcher verhindert, dass der Attention-Mechanismus auf zukünftige Wörter achtet. Damit wird sichergestellt, dass die Vorhersage des nächsten Wortes nur von den bereits erzeugten Wörtern abhängt.

Position-wise Feed Forward

Zusätzlich zu den Attention-Mechanismen existiert in jedem Encoder und Decoder jeweils ein Feed-Forward Netzwerk, welches jede Position (jedes Token oder Wort) in der

Sequenz einzeln verarbeitet, um „den Kontext an jeder Position in eine übergeordnete Repräsentation zu enkodieren“ [90]. Dies ist notwendig für eine gute Leistung von Transformern, da reine Self-Attention-Schichten zu einem Rangverlust („rank collapse“) führen, wodurch ein „token-uniformity inductive bias“ entsteht [87].

Residual Connections, Layer Normalization und Softmax

Innerhalb der Encoder und Decoder, jeweils nach den Attention-Mechanismen und Feed Forward-Netzwerken, werden Additions- („Residual Connections“) [65] und Normalisierungstechniken („Layer Normalization“) [8] eingesetzt. Eine Kombination aus Residual Connections und Layer Normalization kann zur Verbesserung des Trainings und der Generalisierung in tiefen Netzen beitragen und mitigiert das „vanishing and exploding gradient“-Problem.
Schließlich wird die Decoderausgabe mittels einer Lineartransformation und Softmaxfunktion konvertiert, indem eine Wahrscheinlichkeitsverteilung über das Ausgabevokabular erzeugt wird und dem Modell dadurch ermöglicht wird, Vorhersagen zu treffen.

2.3.2 Architekturen von LLMs

Seit der Vorstellung der Transformer-Architektur haben sich verschiedene Architekturen für LLMs etabliert. Die drei Hauptarchitekturen sind dabei die Sequence-to-Sequence-, Autoencoding- und Autoregressive Modelle [138, 172].

Sequence-to-Sequence Modelle

Sequence-to-Sequence Modelle basieren auf der Kombination eines Encoders mit einem Decoder. Grundlage schaffte die Originalarchitektur von Vaswani et al. [158] zu den Transformern. Diese Modelle verarbeiten Eingangsdaten mit dem Encoder und komprimieren diese in eine kontextreiche Repräsentation. Der Decoder generiert anschließend die Ausgabesequenz aus dieser Repräsentation. Diese Modelle sind besonders gut darin, Eingangssequenzen auf Embeddings fester Länge abzubilden, was dem Decoder ermöglicht, kontextuell relevante Ausgaben zu erzeugen. Aus diesem Grund lassen sich Sequence-to-Sequence Modelle vor allem für NLP-Aufgaben wie Textzusammenfassungen, Übersetzungen und Frage-Antwort-Aufgaben nutzen. Durch die Verbindung des Encoders mit dem Decoder entsteht der Nachteil der Erhöhung der Parameteranzahl, was die Effizienz

beeinträchtigen kann. Zudem erfordert das Training dieser Modelle aufgrund der komplexen Angleichung von Eingabe- und Ausgabesequenzen erhebliche Rechenressourcen [138, 172].

Ein Beispiel eines Sequence-to-Sequence LLMs ist die T5-Modellgruppe [123], welche alle NLP-Aufgaben als Text-zu-Text-Transformationen behandelt. Diese Vereinheitlichung ermöglicht es, verschiedene Aufgaben wie Übersetzung, Zusammenfassung und Fragebeantwortung mit einem einzigen Modell durch Trigger als Initialeingabe wie „translate:“, „summarize:“ oder „question:“ zu bearbeiten.

Autoencoding Modelle

Autoencoding Modelle nutzen nur den Encoder-Teil der ursprünglichen Transformer-Architektur, wodurch sie auch als „Encoder-only“-Modelle bezeichnet werden können. Sie verwenden das Trainingsparadigma des „Masked Language Modeling“, bei dem maskierte Wörter innerhalb eines Satzes unter Berücksichtigung des umgebenden Kontexts vorhergesagt werden. In Encoder-Modellen sind alle Eingabetoken durch den Attention-Mechanismus füreinander sichtbar. Diese Modelle eignen sich primär für NLP-Aufgaben, die auf dem Verständnis von Sprache basieren, wobei bidirektionales Lernen und Maskierung zu einem kontextuellen Verständnis führen. Eingeschränkt werden sie jedoch durch die Beschränkung auf Eingabesequenzen fester Länge, die inhärente Kontextabhängigkeit, die die Textgenerierung behindern kann, sowie die Notwendigkeit des Fine-Tunings zur Anpassung an nachgelagerte Aufgaben aufgrund des fehlenden Decoders [138, 172].

Als Beispiel eines autoencoding Modells dient das BERT-Modell [40], das durch seine bidirektionale Verarbeitung des Kontexts in Texten ein tieferes Verständnis der Sprache ermöglicht. Durch das Vortraining mit einem umfangreichen Korpus und eines Fine-Tunings auf verschiedene NLP-Aufgaben hat BERT ein breites Wissen über Sprache erworben, wodurch das Modell an spezifische Aufgaben wie Textzusammenfassungen, Textklassifizierung oder Fragebeantwortung angepasst werden konnte.

Autoregressive Modelle

Autoregressive Modelle zeichnen sich dadurch aus, dass sie hauptsächlich den Decoder-Teil der ursprünglichen Transformer-Architektur nutzen, wodurch sie auch als „Decoder-only“-Modelle bekannt sind. Ihre autoregressive Architektur basiert auf Tokengenerie-

rung, die von vorhergehenden Token abhängt, d.h. Token in diesen Modellen können nur auf Token zugreifen, die ihnen in der Eingabesequenz vorausgehen, wodurch die kausale Struktur der Sprache eingehalten wird. Dadurch eignen sich diese Modelle vor allem für Textgenerierungsaufgaben und bieten Flexibilität bei der Verarbeitung variabler Eingabelängen. Außerdem zeichnen sie sich dadurch aus, dass sie NLP-Aufgaben ohne spezifisches Fine-Tuning mittels Zero- oder Few-Shot-Lernens lösen können [23]. Jedoch kann der Gesamtkontext nur anhand vorhergehender Token erschlossen werden [138, 172].

Zu den autoregressiven Modellen gehört beispielsweise GPT-2 [122], welches das nächste Wort bzw. Token in einer Sequenz vorhersagt und Zero-Shot-Fähigkeiten in verschiedenen NLP-Aufgaben demonstriert hat. Es wurde auf einem umfangreichen Datensatz vortrainiert und war in der Lage, menschenähnlichen Text zu generieren, obwohl ein Fine-Tuning speziell für bestimmte NLP-Aufgaben nicht stattfand. Das Nachfolgemodell GPT-3 [23] zeigte sogar noch bessere Ergebnisse bei Zero- und Few-Shot Aufgaben.

2.3.3 Training und Fine-Tuning von LLMs

Um LLMs einsetzen zu können, müssen sie im Vorfeld mit umfangreichen Datensätzen und Korpora vortrainiert werden. Für nachgelagerte, spezifische Aufgaben oder für bestimmte Verhaltensweisen können sie zusätzlich mittels Fine-Tuning angepasst werden

(Pre-) Training

Das Training bzw. Pre-Training eines LLMs bezeichnet den Schritt, welcher dem Modell ein allgemeines Sprachverständnis sowie kohärente Sprachgenerierung durch Erlernen von Sprachmustern, wie beispielsweise Grammatik oder Semantik, und Weltwissen beibringt [138, 172]. Die Trainingsmethode, die dafür allermeist genutzt wird, bedient sich des Self-Supervised Learnings, bei dem der Trainingsdatensatz ohne Labels auskommt, da die natürliche Abfolge der Wörter als ihre eigene Überwachung („Supervision“) dient. Das Modell wird darauf trainiert, das nächste Wort in einem Text vorherzusagen und lernt, indem es den Fehler bei der Vorhersage des richtigen nächsten Wortes minimiert [77, S. 210].

Um dieses Training für LLMs zu ermöglichen, werden große, umfangreiche Datensätze benötigt. Diese setzen sich aus unterschiedlichen Quellen zusammen, darunter Webseiten aus Crawlingprozessen oder frei verfügbaren Datensätzen. Weiterhin durchlaufen die

Daten vor dem Trainingsprozess verschiedene Filter zur Qualitätssicherung und zur Erfüllung von Sicherheitskriterien. [77, S. 211-213]. Zusätzlich können beim Trainingsprozess verschiedenen Trainingsstrategien auf die Daten angewandt werden, z.B. Trainingsdatenreduktion, neuronale Architektursuche, progressives Lernen oder „mixed precision“ Training [138]. Die Anzahl der Trainingstoken sollte dabei mit der Anzahl an Modellparametern skalieren, um bessere Ergebnisse mit den Modellen erzielen zu können [68].

Fine-Tuning (Post-Training)

Fine-Tuning, oder auch Post-Training, bezeichnet einen Ansatz, um vortrainierte LLMs zur Leistungsverbesserung auf spezifische Aufgaben oder Domänen zu spezialisieren [138]. Diese Ansätze können sich in verschiedenen Ausprägungen äußern, die sich u.a. darin unterscheiden, welche Modellparameter aktualisiert werden. Darunter gehören beispielsweise „Continued Pre-Training“, „Parameter-efficient Fine-Tuning (PEFT)“ und „Supervised Fine-Tuning“ [77, S. 213-214]:
Continued bzw. Continual Pre-Training bezeichnet den Prozess bereits vortrainierte LLMs inkrementell mit neuen Daten weiterzutrainieren, entweder um ihre allgemeinen Fähigkeiten aufrechtzuerhalten („continual general pre-training“) oder um sie an neue Domänen anzupassen („continual domain-adaptive pre-training“). Mit dieser Methode werden alle Parameter des Modells neu trainiert [78].
PEFT ist ein Ansatz, bei dem nur ein kleiner Teil der Modellparameter optimiert wird, während die übrigen Parameter fixiert bleiben, um Rechen- und Speicherkosten drastisch zu senken und trotzdem eine effektive Anpassung von vortrainierte LLMs zu ermöglichen [44]. Mit diesem Ansatz können LLMs auch trainiert werden, um als Klassifizierer oder „Labeler“ für spezifische Aufgaben, wie beispielsweise Sentimentanalysen, zu dienen.
Supervised Fine-Tuning, mit der oftmals „Instruction Fine-Tuning“ umgesetzt wird, ist eine entscheidende Technik, um die Fähigkeiten und Kontrollierbarkeit von LLMs zu verbessern, indem sie auf einem Datensatz, z.B. zur Anweisungsbefolgung, in einer überwachten Weise weiter trainiert werden. Dies schließe die Lücke bei LLMs zwischen dem Vorhersagen des nächsten Wortes und dem Ziel z.B. menschlichen Anweisungen zu folgen [176]. Allerdings reicht das reine Vortrainieren oft nicht aus, um LLMs ausreichend hilfsbereit und sicher zu gestalten. Neben der Anweisungsbefolgung beispielsweise ist auch die Vermeidung von schädlichen oder ungenauen Ausgaben ein zentrales Ziel. Daher wird z.B. Instruction Fine-Tuning häufig mit „Preference Alignment“ kombiniert. Dabei wird ein separates Modell trainiert, um die Übereinstimmung der Antworten eines LLMs mit

menschlichen Präferenzen zu bewerten. Diese Kombination aus Instruction Fine-Tuning und Preference Alignment wird als „Model Alignment“ bezeichnet, mit dem Ziel, die Lernziele der Modelle an die Bedürfnisse und Werte der Menschen anzupassen [77, S. 242].

2.3.4 Retrieval-Augmented Generation mit LLMs

Eine weitere Möglichkeit, LLMs neue Informationen zur Verfügung zu stellen, ist die Technik RAG. Wie Brown et al. zur Vorstellung von GPT-3 gezeigt haben, sind LLMs in der Lage aus neuen Informationen innerhalb ihres Kontextes zu lernen und dieses Wissen auf neue Aufgabenfelder anzuwenden [23]. Diese Form des Lernens wird auch als ICL bezeichnet und wird nach Dong et al. folgendermaßen formalisiert:

> „ICL ist ein Paradigma, das es Sprachmodellen ermöglicht, Aufgaben zu erlernen, wobei nur wenige Beispiele in Form von Demonstrationen gegeben werden.“ [45]

Dieses Paradigma bietet zum einen die Möglichkeit zur Formulierung der Beispiele in natürlicher Sprache und damit eine interpretierbare Schnittstelle zur Interaktion mit LLMs. Dadurch wird die Integration menschlichen Wissens durch Anpassung der Beispiele erleichtert. Zum anderen stellt ICL eine Nachahmung der menschlichen Entscheidungsprozesse dar, bei denen anhand von Analogiebeispielen gelernt wird. Schließlich ist ICL ein trainingsfreier Ansatz im Vergleich zum Supervised Learning, der sowohl Rechenkosten für die Aufgabenadaption reduziert, als auch die Realisierung von „Language-Model-as-a-Service“ fördert und den Einsatz in umfangreichen Anwendungen ermöglicht [45]. Die Technik, Demonstrationen bzw. Beispiele in den Kontext einer Anfrage bzw. eines Prompts an LLMs hinzuzufügen, wird auch als „Few-Shot Prompting“ bezeichnet [77, S. 246].

Mit RAG ist es möglich, vortrainierte parametrische Modelle, wie LLMs, mit einem nicht-parametrischen Speicher zu erweitern, wodurch Probleme wie die Inflexibilität und Interpretierbarkeit von rein parametrischen Modellen adressiert werden können. Diese hybriden Modelle, die parametrisches Wissen mit nicht-parametrischen, retrieval-basierten Speichern kombinieren, ermöglichen es, Wissen direkt, z.B. durch ICL, zu überarbeiten und zu erweitern, wodurch die Ausgaben kontextbezogener, faktisch genauer und vielfältiger sind. Zudem erlaubt der Zugriff auf externes Wissen die Inspektion und Interpretation der Entscheidungsfindung des Modells [84]. RAG ist weiterhin auf eine Vielzahl von

Aufgaben anwendbar und kann verschiedene Arten von externen Wissensquellen nutzen [56]. Dazu bedienen sich RAG-Systeme der Methodiken aus dem IR, um die relevanten Informationen aus dem nicht-parametrischen Speicher zu erhalten.

Information Retrieval

IR ist die Aufgabe, Dokumente auf Grundlage des Informationsbedürfnisses eines Benutzers auf Basis einer Anfrage zurückzuliefern. Dies stellt einen Prozess dar, der auch als „ad-hoc retrieval“ bezeichnet wird. Daraus resultierende Systeme können auch Suchmaschinen genannt werden [77, S. 290-291].
Die Suche und der Abgleich von Anfrage und Zieldokumenten kann über verschiedene Algorithmen durchgeführt werden. Eine Möglichkeit ist der Abgleich auf Basis von Schlüsselwörtern bzw. „Keywords“, die auch als lexikalische Suchalgorithmen bezeichnet werden können. Darunter gehören z.B. tf-idf [125] oder die angepasste Variante BM25 [126], die zur Gewichtung von Begriffen und der daraus resultierenden Bewertung („Scoring“) von Dokumenten anhand ihrer Relevanz verwendet werden.
Eine weitere Möglichkeit ist der Abgleich auf Grundlage semantischer Ähnlichkeit zwischen Anfrage und Dokumenten mit Embeddings, bei dem die Ähnlichkeit mit dem Kosinus zweier Vektoren berechnet werden kann. Dafür können Anfrage und Dokumente über Autoencoding Modelle, wie z.B. BERT, in Embeddings umgesetzt werden, um sie miteinander vergleichen zu können [77, S. 291-298]. Die Suche kann dabei über die Algorithmen „k-nearest neighbor“ oder „approximate-nearest neighbor“ weiter verbessert werden [18].

Um die Suche und den Abgleich zu verbessern, ist das Segmentieren von Dokumenten („Chunking“) ebenfalls von Bedeutung. Chunking ist entscheidend für präzisere Suchergebnisse und die Vermeidung von Kontextbeschränkungen in LLMs. Dazu sind verschiedene Granularitätsstufen umsetzbar, die von Token- über Satz- bis hin zu semantischen Ebenen reichen. Wichtige Aspekte dabei sind die Chunkgröße, Chunking-Techniken und die Anreicherung der Chunks mit Metadaten wie Titeln, Schlüsselwörtern und hypothetischen Fragen, um die Suche weiter zu verbessern [165].

RAG-Architekturen mit LLMs

Grundsätzlich enthalten RAG-Architekturen jeweils eine Retriever- und eine Reader-Komponente, wobei der Retriever darauf ausgelegt ist, aus einem größeren Dokumenten-

bestand relevante Informationen zu einer gegebenen Frage auszuwählen und abzurufen, während der Reader diese abgerufenen Informationen genauer analysiert, um die präzise Antwort auf die Frage zu extrahieren [33]. Im Falle von RAG-Systemen mit LLMs werden beim Retriever relevante Dokumente durch lexikalische und/oder semantische Suche ausgewählt. Anschließend wird mit dem Reader bzw. dem Generator auf Basis der Dokumente eine Antwort generiert [77, S. 301-302], siehe auch Abbildung 2.8.

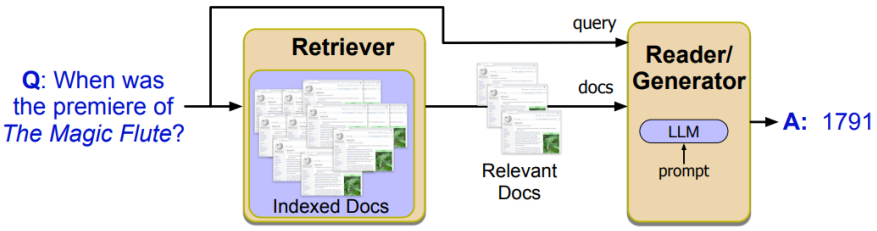

Abbildung 2.8: Retriever-Reader Phasen zur Fragebeantwortung [77, S. 302]

Die Umsetzung von RAG-Systemen mit LLMs ist auf unterschiedliche Arten möglich und kann in drei Paradigmen aufgeteilt werden: naive RAG, erweiterte RAG und modulare RAG [56].
Naive RAG-Implementierungen waren die ersten Umsetzungsformen. Der Prozess besteht dabei hauptsächlich aus der Dokumentindexierung, dem Retrieval und der Generierung und folgt dem einfachen Retriever-Reader Prozess.
Erweiterte RAG verbessert die Qualität des Abrufs durch Optimierung im Pre- und Post-Retrievalprozess. Im Pre-Retrieval können Techniken wie „Query Routing“, „Query Rewriting“ oder „Query Expansion“ eingesetzt werden, um die Anfrage zu den optimalen Kanälen zu führen, sie anzupassen oder in verschiedene Anfragen aufzuteilen und zu erweitern. Im Post-Retrieval können weitere Techniken zur Optimierung erfolgen, wie z.B. „Reranking“, „Summarization“ oder „Fusion“, bei denen die Dokumentreihenfolge optimiert, die Dokumente zur verbesserten Verarbeitung zusammengefasst oder das Scoring vereinheitlicht wird.
Modulare RAG ist das flexibelste Paradigma. Es verwendet statt des klassischen linearen Retriever-Reader-Prozesses, spezialisierte Module, wie z. B. Routing, Such- oder Speichermodule, um die Abruf- und Verarbeitungsfunktionen zu verbessern. Dadurch zeichnet es sich durch gesteigerte Anpassungsfähigkeit und Vielseitigkeit aus [56].

2.4 Stand der Forschung

Bei der Entwicklung von Maschinenübersetzungen wurden in den letzten Jahrzehnten große Fortschritte erzielt. Dabei wurden verschiedene Architekturen und Ansätze entwickelt, die eine immer präzisere Übersetzung von Texten in verschiedenen Sprachen ermöglichen [113, 157, 167, 174, 181]. Inzwischen werden vermehrt neuronale Methoden, insbesondere auf Basis von Transformer-Netzwerken, eingesetzt, die vielversprechende Ergebnisse für monolinguale Maschinenübersetzungen erzielt haben [2, 16, 99]. Auch innerhalb der deutschen Sprache lassen sich Arbeiten finden, die von regelbasierten [148] bis hin zu neuronalen Methoden [130] reichen.
Die monolinguale Maschinenübersetzung für Texte von Standardsprache in Leichter Sprache stellt dabei auf mehreren Ebenen eine Herausforderung dar. Zunächst fehlt eine klare Standardisierung der Leichten Sprache in Deutschland. Neben den inoffiziellen Regelwerken und den wenigen Vorgaben der BITV, die ohne konkrete Umsetzungsbeispiele formuliert sind, existierte diesbezüglich ein erster Entwurf eines Projekts [43], welches vom Bundesministerium für Arbeit und Soziales gefördert wurde und in Form einer DIN SPEC umgesetzt wurde [42]. Die DIN SPEC 33429:2025-03 mit dem Titel „Empfehlungen für Deutsche Leichte Sprache“ beschreibt dabei in 60 Seiten Anwendungsempfehlungen zur Umsetzung von Texten in Leichte Sprache und soll dabei helfen, „gesetzlichen Anforderungen zur Verwendung von Leichter Sprache zu präzisieren“ [42, S. 9]. Die Leichte Sprache selbst kann zudem als „Low-Resource“-Sprache bezeichnet werden. Low-Resource Sprachen sind dadurch gekennzeichnet, dass sie „u.a. als weniger erforscht, ressourcenarm, weniger computerisiert, weniger privilegiert, weniger häufig gelehrt oder von geringer Dichte charakterisiert werden“ [97]. Dies trifft auch auf die Leichte Sprache zu und zeigt sich anhand der wenigen verfügbaren parallelen Korpora. Einige Forschungsarbeiten haben sich diesbezüglich primär mit der Erstellung von parallelen Korpora für die Leichte Sprache beschäftigt [46, 133, 145], jedoch sind Datensätze für den behördlichen Kontext nur wenige bis gar keine vorhanden. Dazu erstellte Madina et al. [96] beispielsweise eine Übersicht zu unterschiedlichen Korpora für die Leichte Sprache. Diese wiesen jedoch Inkonsistenzen auf, da sie nur teilweise parallel, vergleichbar oder wenig bis gar nicht aligned seien, aber unterschiedliche Komplexitätsgrade aufweisen. Zusätzlich erschwere die inkonsistente Verfügbarkeit vereinfachter Inhalte die Erstellung paralleler, satzweise aligneter Korpora. Zu einem ähnlichen Schluss kamen Stodden et al. [145]. Sie stellten fest, dass viele Datensätze für das Training von Modellen zur Textvereinfachung nur eingeschränkt nutzbar sind, da sie beispielsweise zu klein, qualitativ fragwürdig aligned, rein evaluativ, nicht parallel oder schlichtweg nicht verfügbar sind, auch teils aufgrund von

Urheberrechtsbeschränkungen. Um diese Einschränkungen zu mitigieren, erstellten sie in der vorgestellten Arbeit verschiedene parallele Korpora. Durch den Mangel an parallelen Korpora wurden zudem Ansätze entwickelt, die diesen Mangel durch Fine-Tuning von Sprachmodellen beheben sollen. Anschütz et al. [5] zeigten beispielsweise eine zweistufige Methode, bei der zuerst durch ein Fine-Tuning autoregressive Modelle angepasst und diese dann in einem Sequence-to-Sequence-Modell eingebettet wurden. Einen weiteren Fine-Tuning-Ansatz beschreibt Klöser et al. [81], bei der ein paralleles Korpus mithilfe synthetischer Daten generiert und zum Fine-Tuning genutzt wurde. Dazu wurden LLMs genutzt, um aus bereits vereinfachten Textsegmenten von Webseiten, die möglichen korrespondierenden Texte in Standardsprache zu erhalten. Dieses semi-synthetische parallele Korpus wurde schließlich für das Fine-Tuning genutzt.
Um ein ressourcenschonendes Fine-Tuning zu ermöglichen, können PEFT-Techniken wie Low-Rank Adaptation (LoRA) [70] für LLMs genutzt werden. Dazu wurden verschiedene Ansätze von PEFT-Techniken zur Maschinenübersetzung von Low-Resource Sprachen untersucht [147], wobei Liang et al. [86] aufzeigten, dass durch Einsatz von LoRA für Low-Resource Sprachen gleichwertige oder gar bessere Ergebnisse erzielt werden können, trotz der Knappheit von Trainingsdaten. Zwar bleibt die Leistung des LoRA-Ansatzes auf lange Sicht hinter beispielsweise Continued Pre-Training-Ansätzen, jedoch bietet sie einen Vorteil hinsichtlich Speicher- und Recheneffizienz, wodurch auch kleinere Modelle konkurrenzfähige Ergebnisse erzielen und insbesondere in ressourcenbeschränkten Umgebungen Einsatz finden könnten.
Auch RAG in Verbindung mit ICL findet hinsichtlich Low-Resource Sprachen Anwendung. So zeigen Nazi et al. [108] und Zhu et al. [180] auf, dass ICL-Techniken, wie beispielsweise Few-Shot Prompting, in Bezug auf verschiedene Aufgabenstellungen der Low-Resource Sprachen die Qualität der Antworten von LLMs erhöhen können. Der Einsatz von RAG in Kombination mit ICL lässt sich auch im Bereich der neuronalen Maschinenübersetzungen finden, welche den Ansatz des TM nutzen [31, 66, 175], und mit LLMs kombinieren [163]. Es zeigte sich, dass aus dieser Kombination die Übersetzungsqualität auch signifikant gesteigert wird. Auch in weiteren konkreten Beispielen für andere Low-Ressource Sprachen zeigt sich dies [15, 69, 100, 109]. Der Einsatz einer hybriden Methode aus beispielbasierten- (TM) und neuronalen Methoden (LLMs) wird auch bei Kopp et al. [83] als valide Möglichkeit angesehen, um im deutschen öffentlichen Sektor Standardsprache in Leichte Sprache zu übersetzen. Kopp et al. schlagen dabei den Einsatz eines TM für den öffentlichen Sektor vor, um einerseits sofortigen Mehrwert im Übersetzungsprozess zu generieren, indem passende Textfragmente zur Wiederverwendung oder Inspiration bereitgestellt werden. Andererseits könne ein TM langfristig auch

als Trainingsmaterial für Maschinenübersetzungssysteme auf Basis neuronaler Methoden dienen. Der Einsatz eines TMs in Kombination mit LLMs wurde ebenfalls von Mu et al. [106] untersucht. Sie konnten eine verbesserte Übersetzungsqualität durch ICL im Gegensatz zur einfachen Nutzung von LLMs feststellen. Zur Erstellung eines parallelen Korpus für Leichte Sprache, welches als TM dienen könnte, können neben synthetischen Methoden auch vorhandene Datensätze über Sentencealignment-Algorithmen aligned werden, die Spring et al. untersuchten [143]. Sie stellten fest, dass der Large-Scale Hierarchical Alignment (LHA)-Algorithmus [115] insgesamt mit die besten Sentencealignments auf deutsche Texte ermöglicht. Auch Sentencealignments durch Nutzung von autoregressiven Modellen und entsprechenden Anweisungen bzw. Prompts wurden untersucht, welche ebenfalls gute Ergebnisse lieferten [85].
Bei der Frage, welcher Ansatz zwischen Fine-Tuning und Nutzung von RAG mit ICL die besseren Resultate liefert, lässt sich festhalten, dass beide Ansätze die Ergebnisse signifikant verbessern und dieser Effekt sich verstärken könnte, wenn sie gemeinsam genutzt werden. Werden sie aber jeweils einzeln genutzt, sind Ansätze mit RAG und ICL gegenüber Fine-Tuning besser geeignet [10, 117, 141]. Jedoch zeigten Liu et al. [88], dass PEFT gegenüber ICL besser und recheneffizienter sein kann. Diese Studien wurden jedoch nicht in Bezug auf die Übersetzungsdomäne erstellt, wodurch diese nur Indizien für mögliche Verbesserungspotenziale darstellen können.
Bei der Evaluation der Textvereinfachung sollte beachtet werden, dass beim Entfernen von Inhalten aus einem Satz zwar die Einfachheit erhöht, aber auch die Semantik reduziert wird und diese beiden Aspekte negativ korrelieren. Dies müsse nach Schwarzer und Kauchak [135] bei Systemvergleichen zu Metriken der Einfachheit, Semantik und Grammatikalität beachtet werden. Eine Verbesserung der einen Metrik ohne Verbesserung einer anderen lasse sich auf diese Beziehung zurückführen und nicht nur aufgrund der Systemleistung. Nach einer Metastudie von Alfear et al. [1] sei LENS [95] zudem eine der besten referenzbasierten Metriken, um Sätze nach Einfachheit, Semantik und Grammatikalität zu evaluieren. Alternativ sei BERTScore ebenfalls geeignet. BLEU und SARI seien zwar weniger geeignet, jedoch sollten diese als Vergleichswerte in neueren Publikationen gegenüber älteren dennoch berücksichtigt werden.
Abschließend zeigten Asghari et al. auf, dass „die überwiegende Mehrheit der Webseiten des öffentlichen Sektors [...] noch immer nicht barrierefrei ist“ [7] hinsichtlich der Verfügbarkeit der Leichten Sprache. Zudem deutete Feedback an Asghari et al. von öffentlichen Stellen darauf hin, dass zwar Prozesse zur Erstellung von Inhalten in Leichter Sprache existieren, die Kosten jedoch erheblich variieren können. Diese reichen von geschätzten 140-250 Euro für die Prüfung und Zertifizierung bereits übersetzter Inhalte durch externe

Dienstleister bis hin zu 4900 Euro für die vollständige Analyse und Übersetzung einer Webseite.

3 Konzeption und Implementation

Zur Untersuchung der verschiedenen Optimierungsansätze zur Erstellung von monolingualen Maschinenübersetzungen von Texten aus Standardsprache in die Leichte Sprache, wurde in dieser Arbeit eine technische Umsetzung gewählt, die dem allgemeinen Workflow nach Anisuzzaman [4] bzw. dem Prozessmodell, siehe Abbildung 3.1, folgt. Nach dem Workflow bzw. dem Prozessmodell von Anisuzzaman lasse sich das Fine-Tuning von LLMs für „spezielle Anwendungsfälle“ in einem Prozess aus Datenaufbereitung, Modellauswahl, Fine-Tuning und Validierung darstellen.

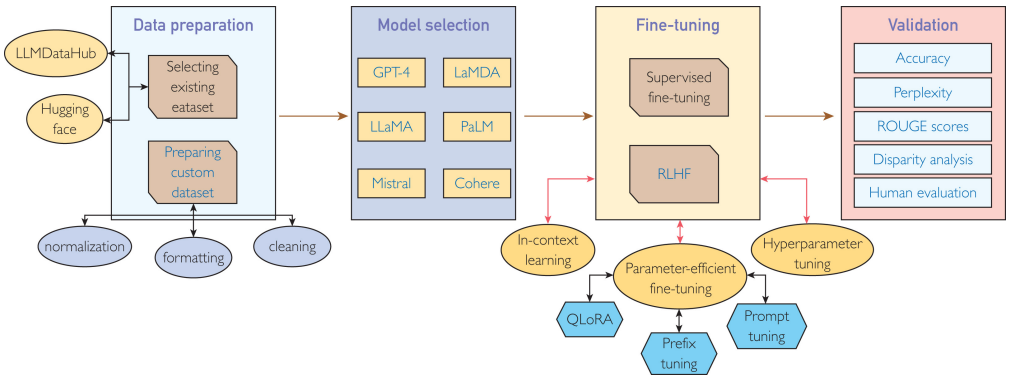

Abbildung 3.1: Allgemeiner Workflow des LLM Fine-Tunings [4]

Im ersten Schritt, bei der Datenaufbereitung, könne auf zwei verschiedene Quellen zugegriffen werden: Auf bereits existierende Datensätze über bestimmte Webseiten, wie z.B. Huggingface[1], oder eigens erstellte Datensätze, die entsprechend für die weitere Verarbeitung, durch z.B. Datenbereinigung („Cleaning“), aufbereitet werden müssen.
Auf Basis der vorliegenden Daten und der Rahmenbedingungen des Anwendungsfalles, spielt die Auswahl eines geeigneten LLMs eine entscheidende Rolle. Zu den Kriterien für die Auswahl der Modelle gehören dabei die spezifische Aufgabe, die das Modell durchführen soll, die erwartete Leistung, die Ein- und Ausgabeanforderungen sowie generelle Modelleigenschaften wie die Parameterzahl. Die Modellarchitektur kann ebenfalls eine

[1] https://huggingface.co - Abgerufen am 16.06.2025

Rolle spielen.
Beim Fine-Tuning werden die ausgewählten Modelle im nächsten Schritt an die spezifischen Aufgaben adaptiert. Zum Fine-Tuning gehören nach Anisuzzaman neben dem Supervised Fine-Tuning oder dem Reinforcement Learning from Human Feedback (RLHF), die Teile des Preference Alignments sind, auch Methoden wie ICL oder PEFT, welche unter anderem LoRA-Ansätze inkludieren. Zusätzlich kann auch eine Hyperparameteroptimierung der Modelle erfolgen.
Schließlich erfolgt im letzten Schritt die Validierung bzw. Evaluation der Ergebnisse, die sich aufgrund des generativen Charakters der LLMs komplex gestalte, da im Gegensatz zur prädiktiven Künstliche Intelligenz (KI) die generative KI neue Inhalte erzeugt, die schwieriger zu bewerten sei. Generell lassen sich zwei verschiedene Evaluationsmöglichkeiten unterscheiden. Bei der internen Validierung erfolgt die Auswahl des optimalen Modells durch Überwachung des Lernprozesses und Implementierung von Abbruchkriterien des Trainings. Andererseits kann ein Evaluationsdatensatz auf Basis echter Daten zur Überprüfung der Modellleistungen in Betracht gezogen werden. Zusätzlich können zentrale Leistungsmetriken, wie z.B. Genauigkeit, Perplexität oder auch menschliche Evaluation genutzt werden [4].
Für die vorliegende Arbeit wurde im Datenaufbereitungsschritt aufgrund der genannten Herausforderungen, siehe dazu Kapitel 2.4, bei den Datensätzen für Leichte Sprache, speziell im behördlichen Kontext, der Weg eines eigens erstellten Datensatzes durch Web Scraping gewählt, welcher mit synthetischen Daten weiter angereichert wurde. Die Modellauswahl beschränkte sich aufgrund der ressourcenbeschränkten Umgebung im behördlichen Kontext auf kleinere Sprachmodelle, die im weiteren Verlauf zum Leistungsabgleich mit SOTA Modellen verglichen wurden. Innerhalb des Fine-Tuning Prozesses wurden die ausgewählten Modelle darauf ausgerichtet einerseits mit einem eigens entwickelten RAG-System zur Umsetzung der ICL-Technik, andererseits auf Basis der PEFT-Technik Quantized Low-Rank Adaptation (QLoRA) Texte in Leichter Sprache zu erstellen. Zusätzlich wurde ebenfalls die Auswirkung der Kombination beider Techniken, ICL und Fine-Tuning, auf die Generierung der Texte untersucht. Im Gegensatz zu Anisuzzaman wird in dieser Arbeit jedoch zwischen dem Einsatz von ICL und dem Fine-Tuning der Modelle unterschieden. Diese werden aber auf gleicher Prozessebene innerhalb des Workflows angesehen. Bei der abschließenden Evaluation wurden die generierten Texte der ausgewählten Basismodelle ohne Nutzung der Techniken, mit Nutzung von Fine-Tuning bzw. des RAG-Systems für das ICL sowie die Kombination aus beiden ausgewertet. Zum Abgleich der Basismodellleistung bei der Generierung wurden ausgewählte SOTA-Modelle ebenfalls berücksichtigt. Die Auswertung erfolgte auf Basis verschiedener automatisier-

ter Metriken, die die Kriterien zur Bewertung der Leichten Sprache bzw. der monolingualen Maschinenübersetzung hinsichtlich Einfachheit, Grammatikalität und Semantik berücksichtigen, um eine umfassendere Bewertung zu ermöglichen. Obwohl die Kriterien Einfachheit, Grammatikalität und Semantik primär durch manuelle Evaluationen bewertet werden, können sie auch durch ausgewählte automatisierte Metriken weitgehend abgedeckt werden, siehe auch Kapitel 2.2.3. Abschließend wurde zur Erstellung der Code-Artefakte die Programmiersprache Python genutzt.

Das für diese Arbeit implementierte System zur Generierung der Übersetzungen lässt sich in Abbildung 3.2 zur Übersicht einsehen. Der Prozess bei der Nutzung des Systems läuft dabei in fünf Schritten ab: 1. Der User schickt seine Anfrage, das Dokument, an das System. 2. Innerhalb des Systems werden durch eine semantische und lexikalische Suche die entsprechenden TUs für jeden Satz des Dokumentes zurückgeliefert und die Ergebnisse miteinander kombiniert. 3. Die Beispiele werden nach ihrer Relevanz zur Anfrage neu geordnet. 4. Die entsprechenden Beispiele werden in einem System-Prompt, in dem die Anfrage enthalten ist, eingebettet, welches als Anweisung an das jeweilige Modell geschickt wird. 5. Das Modell generiert die entsprechende Übersetzung und das System liefert die Antwort an den User zurück.
Für die Evaluation der Modelle ohne RAG wird die Retriever-Komponente des Systems übersprungen und stattdessen nur die Generator-Komponente verwendet. Die genaue Implementation und verschiedenen Einstellungsmöglichkeiten des Systems werden in den folgenden Abschnitten beschrieben.

3.1 Datensatzaufbereitung

Um für das spätere RAG-System und dem Fine-Tuning der Modelle die Datengrundlage zu schaffen, wurde einerseits ein bestehendes paralleles Korpus aus Webseiten extrahiert und andererseits ein paralleles Korpus anhand synthetischer Daten erstellt. Zusätzlich diente eine Submenge des parallelen Korpus aus Webseiten zur Erstellung eines Evaluationsdatensatzes, um die Leistung der verschiedenen Modelle überprüfen zu können. Da in dieser Arbeit untersucht wurde, inwiefern ausgewählte Modelle mithilfe der Optimierungsansätze satzweise Maschinenübersetzungen erstellen können, wurden die Texte aus den Datensätzen im Verlauf durch Sentencealignment jeweils in Form von TUs gebracht.

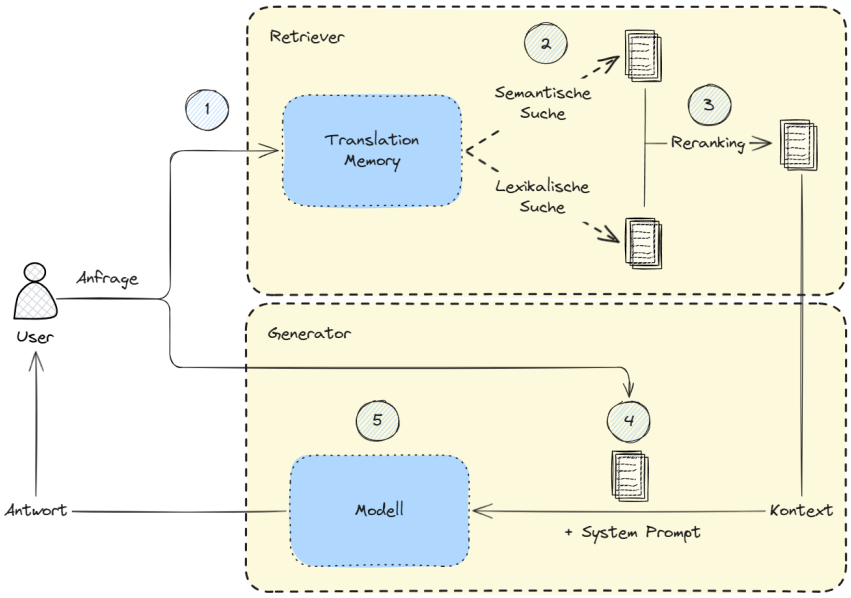

Abbildung 3.2: Übersicht des implementierten RAG-Systems

3.1.1 Datensatzerstellung durch Web Scraping

Zur Erstellung eines ersten Datensatzes wurden zunächst die 16 Webseiten der Bundesländer untersucht und überprüft, ob behördliche Texte extrahiert werden konnten. Da von diesen lediglich drei Bundesländer die Möglichkeit boten, ein paralleles Korpus zu extrahieren, wurde die Suche ausgeweitet auf die einzelnen Webseiten der Landeshauptstädte der Bundesländer, exklusive der Stadtstaaten Berlin, Bremen und Hamburg. Dadurch wurden entsprechend ausreichend Artikel in Standardsprache und Leichte Sprache identifiziert, aus denen ein paralleles Korpus extrahiert werden konnte. Eine genauere Auflistung dieser Funde je Bundesland und Landeshauptstadt lässt sich unter A.1 respektive A.2 finden.
Der Prozess zur Extrahierung gestaltete sich in Form von Web Scrapern, durch die, ausgehend von einer Start-URL, die als Übersicht aller Artikel diente, sich alle verfügbaren Artikel in Standardsprache und Leichte Sprache extrahieren ließen. Zur Erstellung der Scraper wurde die Software Scrapy[2] verwendet, welche ein Open Source Framework für Webcrawling und -scraping ist. Es bietet als Framework einen leichten und schnellen Einstieg zur Erstellung von Crawlern und Scrapern und ist zudem performant bei der

[2] https://www.scrapy.org - Abgerufen am 16.06.2025

Extrahierung der Daten von einzelnen Seiten. Ausgehend von einem Scraping-Template, welches den allgemeinen Aufbau und Funktionsweise aller Scraper beinhaltet, wurden für alle acht Webseiten ein Scraper implementiert, die das Crawlen, Scrapen, aber auch das Parsen der einzelnen Artikel auf der jeweiligen Webseite übernahm. Der Aufbau aller Scraper ist identisch, bei der jeder Scraper mit Start-URls beginnt, mit den entsprechenden Übersichten zu den Artikeln in Leichter Sprache. Zusätzlich existieren jeweils drei Parser Funktionen: `parse`, `parse_article_page` und `parse_normal_page`. Während `parse` als genereller Startpunkt zur Extraktion aller Artikel dient, wird jeder Link zur `parse_article_page`-Funktion weitergeleitet, die die URL des Artikels in Leichte Sprache sowie den eigentlichen Text extrahiert. Es wurde darauf geachtet, dass ausschließlich der Text extrahiert wurde und keine Bilder oder Videos. Zusätzlich wurde für jeden Artikel jeweils der Link zum Artikel in Standardsprache extrahiert, falls ein solcher vorhanden war. Falls keiner vorhanden war, wurde nur die URL und der Text in Leichte Sprache extrahiert. Ansonsten wurde die Funktion `parse_normal_page` aufgerufen, mit der, ähnlich wie die vorherige genannte Funktion, die URL und der Text in Standardsprache extrahiert wurde. Pro Webseite wurde jeweils eine JSON-Datei mit allen Informationen mit den Inhalten `simple_url`, `simple_text`, `normal_url` und `normal_text` erstellt. Schwierigkeiten ergaben sich bei der genauen Lokalisierung der einzelnen Texte, da diese bei jeder Webseite anders eingebettet sind. Entsprechend musste jeder Scraper jeweils individuell angepasst und erweitert werden. Beispielsweise wurden individuell Tags entfernt, die Bilder, Videos oder andere Elemente, wie Gliederungstabellen oder andere Links, enthalten haben. Die Vorlage zur Erstellung der einzelnen Scraper der Bundesländer- und Landeshauptstadtwebseiten, lässt sich unter A.4 einsehen.

Zusammengefasst ließen sich 609 Artikel in Standardsprache und Leichte Sprache aus den verschiedenen Quellwebseiten extrahieren, wobei `hannover.de` die meisten Artikel vorwies, wie in Tabelle 3.1 dargestellt.

Um den Datensatz zu erweitern, wurde der Datensatz mit dem `stadt-koeln.de`-Subset aus [155] komplettiert.

3.1.2 Datensatzerstellung durch Generierung synthetischer Daten

Um den Datensatz aus den Webseiten zu erweitern, wurden zusätzlich synthetische Daten auf Satzbasis generiert. Das daraus resultierende parallele Korpus diente im weiteren Verlauf zusammen mit den parallelen Korpora aus den Webseiten als Basis zur Erstellung des

Quelle	Anzahl der Artikel
hannover.de	272
hamburg.de	103
stadt-koeln.de	82
dresden.de	67
stuttgart.de	44
stadt.muenchen.de	28
hessen.de	5
saarbruecken.de	4
ms.niedersachsen.de	4
Summe	609

Tabelle 3.1: Anzahl der Artikel je Quelle

TMs und für das Fine-Tuning ausgewählter Modelle mit LoRA. Die Generierung der synthetischen Daten erfolgte über die Nutzung des LLMs `Gemini-2.5 Flash Preview` im „Thinking"-Modus von Google über die Plattform OpenRouter[3]. Die Erzeugung von synthetischen Daten mithilfe von LLMs zeigt insbesondere im Bereich von Low-Ressource Umgebungen einen vielversprechenden alternativen Ansatz, um den Mangel an Daten zu mitigieren [64, 89, 120]. OpenRouter ist des Weiteren ein Anbieter, welcher OpenAI-kompatible Completion APIs für über 300 Modelle[4] und damit eine große Flexibilität hinsichtlich der Nutzung verschiedener Modelle anbietet. Das Modell Gemini 2.5 Flash Preview wurde aufgrund seines Preis-Leistungsverhältnisses und der Geschwindigkeit der Generierung der Dokumente ausgewählt[5].
Zur Umsetzung der Datengenerierung wurden die zwei Klassen `LLMClient` und `LLMSyntheticDataGenerator` implementiert. Die Klasse `LLMClient` stellt dabei die Verbindung zu OpenRouter via der OpenAI-kompatiblen API her und definiert eine Funktion zur Generierung von Antworten. Zusätzlich kann im Konstruktor das auszuwählende Modell gewählt werden. Die Klasse `LLMSyntheticDataGenerator` definiert einen System-Prompt als Anweisung zur Generierung synthetischer Daten und eine Funktion, um die Anweisungen via API an die Modelle zu senden. Dazu kann ein spezifisches Thema sowie die Anzahl der generierten Dokumentbeispiele ausgewählt werden, welches dem System-Prompt zugeführt wird. Das Klassendiagramm in Abbildung 3.3 zeigt die Eigenschaften und Beziehung der Klassen untereinander. Weiterhin lassen sich die Im-

[3] https://openrouter.ai - Abgerufen am 16.06.2025
[4] https://openrouter.ai/docs/overview/models - Abgerufen am 16.06.2025
[5] https://deepmind.google/models/gemini/flash/ - Abgerufen am 16.06.2025

plementierungen der beiden Klassen unter A.5 respektive A.6 mitsamt System-Prompt einsehen.

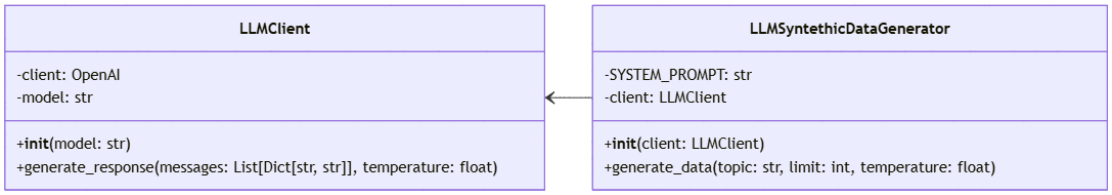

Abbildung 3.3: Klassendiagramm für die Klassen LLMClient und LLMSyntheticDataGenerator

Zur Generierung der Daten via `Gemini-2.5 Flash Preview` im „Thinking"-Modus wurde sich der Structured-Output Funktionalität des Modells bedient, bei der durch die Definition eines `response_format` die Ausgabe im JSON-Format geliefert wird, wodurch eine einfachere Verarbeitung ermöglicht wurde. Dem Modell wurde angewiesen, zunächst jeweils ein Dokument in Standardsprache und Leichte Sprache zu verfassen und die daraus resultierenden Sätze in den Texten eigenständig zu alignen und als JSON-Objekt zurückzusenden. Dadurch konnten die einzelnen Sätze als TUs erstellt werden. Über die Argumente `topic` und `limit` der Funktion `generate_data` konnte zudem das Thema und die Anzahl der erstellten Dokumente für das Sentencealignment ausgewählt werden. Um sicherzustellen, dass die einzelnen TUs ausreichend ähnlich zueinander sind, wurde zum Abgleich ein Embeddingmodell genutzt. Mit diesem wurde die Kosinus-Ähnlichkeit der vektorisierten Texte berechnet, und Texte mit Werten außerhalb des Bereichs von 0.70 und 0.99 wurden herausgefiltert, um den Kriterien des Full-Match bzw. Fuzzy-Match zu genügen [35]. Beispielsweise konnte aus einem Dokument zum Thema Demokratie folgende TUs mit einem Ähnlichkeitswert von 0.72 extrahiert werden:

> **Standardsprache:** Dies stärkt die Demokratie und führt oft zu besseren Lösungen, da unterschiedliche Perspektiven berücksichtigt werden.
>
> **Leichte Sprache:** Das ist wichtig für die Demokratie. So gibt es oft bessere Lösungen. Viele Menschen können ihre Meinung sagen.

Das gewählte Embeddingmodell zur Ermittlung des Ähnlichkeitswerts ist das Modell `jina-embeddings-v2-base-de` des Unternehmens Jina AI. Dieses schneidet nicht nur bei der Leistung im Bereich der deutschen Sprache besser ab als beispielsweise `multilingual-e5-large` [76, 146], sondern bietet auch Modelle mit einem Bruchteil der Parameterzahl größerer Embeddingmodelle an. Diese lassen sich daher auch in

ressourcenbeschränkten Umgebungen besser einsetzen. Zudem ist das gewählte Modell ebenfalls öffentlich zugänglich, da es eine Open-Source Lizenz besitzt. In dieser Arbeit wurde sich aus diesem Grund für das Embeddingmodell `jina-embeddings-v2-base-de` entschieden, da die Leistungsunterschiede zum `jina-embeddings-v3`-Modell im Bereich der Deutschen Sprache auf einem ähnlichen Niveau sind [75, 104]. Zudem ist das erstere Modell mit 161 Millionen Parametern und 322MB[6] vergleichsweise klein gegenüber dem letztgenanntem Modell mit 572 Millionen Parametern und mit 1.14GB Größe[7].
Zur Generierung der TUs wurde eine ausgewählte Liste an Themen verwendet, die in behördlichen Texten behandelt werden könnten, wie z.B. Demokratie, Wohnberechtigungsschein oder Stadtverwaltung. Zu jedem Thema wurden zwei bis drei Dokumente erstellt, deren Sätze über das Modell aligned und als JSON-Objekt zurückgegeben wurden. Insgesamt wurden dadurch ungefiltert 3524 TUs erstellt. Bei Anwendung der Filter zur semantischen Ähnlichkeit zwischen 0.7 und 0.99, konnten 2886 TUs für die weitere Verarbeitung genutzt werden.

3.1.3 Aufbereitung

Für die Nutzung der Optimierungstechniken ICL durch RAG mittels eines TMs sowie Fine-Tuning der Modelle, mussten neben den synthetischen Daten, die bereits in der erforderlichen Form der TUs vorlagen, weiterhin ein Sentencealignment der Dokumente des Web Scraping Datensatzes der Webseiten durchgeführt werden. Dazu wurde in einem ersten Schritt der LHA-Algorithmus angewandt, welcher jedoch unzureichende Ergebnisse auf Basis der vorliegenden Dokumente lieferte. Alternativ wurde der Ansatz gewählt, das Sentencealignment ebenfalls über das `Gemini 2.5 Flash Preview`-Modell umzusetzen. Dies geschah unter der Annahme von Senrich und Volk sowie Thompson und Koehn [137, 153], dass Maschinenübersetzer bzw. Transformer-Modelle ebenfalls in der Lage sind qualitativ hochwertiges Sentencealignment durchzuführen. Analog zur `LLMSynthetic-DataGenerator`-Klasse, wurde die `LLMSentenceAligner`-Klasse implementiert, die ebenfalls ein `LLMClient` als Objekt nutzt, um die Anweisung aus dem System-Prompt über die `create_alignments_from_documents`-Funktion umzusetzen. Ein Klassendiagramm der Beziehung und Eigenschaften der Klassen `LLMClient` und `LLMSentenceAligner` lässt sich auf Abbildung 3.4 sowie die Implementation der Klasse unter A.7 einsehen.

[6] https://huggingface.co/jinaai/jina-embeddings-v2-base-de - Abgerufen am 16.06.2025
[7] https://huggingface.co/jinaai/jina-embeddings-v3 - Abgerufen am 16.06.2025

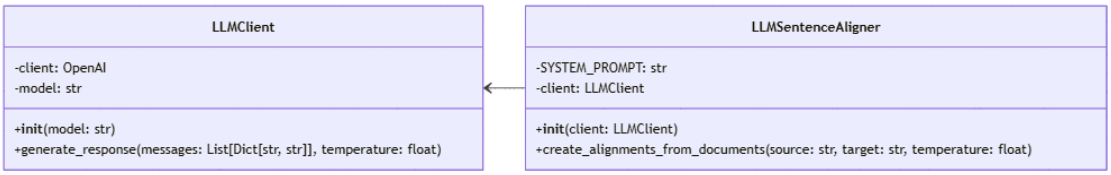

Abbildung 3.4: Klassendiagramm für die Klassen LLMClient und LLMSentenceAligner

Im Zuge des Sentencealignments wurden die Sätze aus 509 Artikeln aus dem Web Scraping Datensatz der Webseiten mithilfe des `LLMSentenceAligner` aligned. Die Differenz von 100 Artikeln zum Ursprungsdatensatz rührt daher, dass zur Erstellung des Evaluationsdatensatzes jene 100 manuell ausgewählte Artikel entfernt wurden. Aus diesen 509 Artikeln wurden insgesamt 3257 TUs mit Ähnlichkeitswerten zwischen 0.70 und 0.99 generiert. Die Ähnlichkeitswerte wurden analog zu den Werten aus der synthetischen Datengenerierung ermittelt.
Die Aufbereitung der 2886 TUs aus dem synthetischen Datensatz erfolgte, indem exakt 500 Beispiele für das Fine-Tuning der Modelle gesampelt und herausgezogen wurden. Die restlichen 2386 TUs wurden zusammen mit den 3257 Beispielen aus dem Web Scraping Datensatz für die Erstellung des TMs kombiniert, wodurch eine Gesamtanzahl von 5643 TUs für das TM zur Verfügung standen. Der Datensatz bestand schlussendlich aus 58 % echten und 42 % synthetischen Daten.
Für das spätere Fine-Tuning musste der Datensatz mit den 500 TUs zudem in ein bestimmtes Format, in Chat-Templates, überführt werden. Zu diesen Chat-Templates gehören unter anderem das Alpaca-Format[8] oder das ShareGPT-Format[9]. In diesem Fall wurde das ShareGPT-Format genutzt, welches besonders gut geeignet ist für Multi-turn Konversationen, jedoch auch für First-turn Konversationen genutzt werden kann. Dazu wurde darauf geachtet, dass der System-Prompt aus den späteren Implementationen mit denen für das Fine-Tuning übereinstimmt und der zu übersetzende Satz ebenfalls in den System-Prompt eingebettet wurde. Die mögliche Antwort in Leichte Sprache wurde abschließend ebenfalls in die Antwort des LLMs in das Template eingebettet. Ein Beispielelement der daraus resultierenden JSON-Datei lässt sich unter A.3 einsehen.
Für die Auswahl der 100 Artikel des Evaluationsdatensatzes sind lediglich die Artikel berücksichtigt worden, die eine Tokenanzahl zwischen 256 und 768 besitzen, damit keine zu kurzen, aber auch keine zu langen Texte für die abschließende Evaluation der Modelle zur Verfügung standen. Weiterhin wurden die Artikel manuell bereinigt.

[8] https://huggingface.co/datasets/tatsu-lab/alpaca - Abgerufen am 16.06.2025
[9] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered - Abgerufen am 16.06.2025

Zusammenfassend sind bei der Datenaufbereitung mehrere Datensätze für unterschiedliche Zwecke entstanden. Neben dem Datensatz bestehend aus 5643 TUs für das TM des RAG-Systems zur Nutzung der ICL-Technik, wurde ein Fine-Tuning Datensatz aus 500 TUs, sowie ein Evaluationsdatensatz bestehend aus 100 Artikeln des Web Scraping Datensatz der Webseiten erstellt.

3.2 Modellauswahl

Zur Auswahl der geeigneten Modelle musste darauf geachtet werden, dass diese in ressourcenbeschränkten Umgebungen im behördlichen Umfeld sowie innerhalb der datenschutzrechtlichen Bestimmungen innerhalb dieses Kontextes genutzt werden können. Entsprechend boten sich in diesem Zusammenhang die Nutzung von Modellen mit überschaubarer Parameteranzahl aus dem Open-Source Bereich an. Grundsätzlich ist hierbei zwischen Open-Source- und „Open-Weight"-Modellen zu unterscheiden. Während Open-Source den vollständigen Zugriff auf Quellcode, Architektur und Trainingsdaten impliziert, beschränkt sich Open-Weight auf die Veröffentlichung der trainierten Modellgewichte. Da für den Anwendungsfall dieser Arbeit die lokale Verfügbarkeit und Anpassbarkeit der Gewichte entscheidend ist, werden die Begriffe Open-Source bzw. quelloffene Modelle in dieser Arbeit vereinfachend auch für Open-Weight-Modelle verwendet.
Kleinere Modelle mit überschaubarer Parameterzahl, oder auch „Small Language Models (SLMs)", zeichnet aus, dass sie aufgrund ihrer Effizienz und ihres minimalen Bedarfs an Rechenressourcen besonders für den Einsatz in ressourcenbeschränkten Umgebungen geeignet sind. Beispielsweise können diese direkt auf Endgeräten („On-Device") oder Edge-Geräten eingesetzt werden. Dies ermöglicht eine geringe Inferenzlatenz, Kosteneffektivität und eine einfache Anpassbarkeit sowie effizienteres Fine-Tuning für spezifische Aufgabenbereiche oder zur Integration von Domänenwissen. Ein wesentlicher Vorteil ist dabei die Möglichkeit der lokalen Datenverarbeitung, was den im behördlichen Kontext kritischen Datenschutzanforderungen entgegenkommt und Bedenken hinsichtlich der Nutzung von Cloud-APIs, wie sie bei größeren Modellen oft notwendig ist, minimiert [91, 114, 161]. In dieser Arbeit werden Modelle mit weniger als zehn Milliarden Parametern als SLMs klassifiziert, da diese Größenordnung typischerweise ein ressourcenschonendes Fine-Tuning auf gängiger Consumer-Hardware und einen lokalen Betrieb auf dieser ermöglicht.
Ein weiteres Auswahlkriterium beinhaltete, dass die Modelle mindestens einem Instruct-basiertem Fine-Tuning unterzogen worden sind. Instruct-Modelle sind angepasste Modelle, die speziell auf die Befolgung von Anweisungen trainiert wurden. Dies ist für die

Nutzung spezifischer Anweisungen im System-Prompt von Bedeutung. Damit können Anweisungen an das Modell gegeben werden, wie der Text von Standardsprache in Leichte Sprache übersetzt werden kann. Abschließend war es von Bedeutung, dass die Modelle ebenfalls zur Nutzung der deutschen Sprache vortrainiert worden sind. Um eine bessere Übersicht über die Leistungsfähigkeit verschiedener Modelle geben zu können, wurden folgende drei Instruct-Modelle für diese Arbeit zur Untersuchung in Betracht gezogen:

- **Llama 3.1 8B[10]:** Dieses Modell von Meta AI ist ein quelloffenes, auf einer Decoder-only Architektur basierendes Sprachmodell mit ca. 8 Milliarden Parametern, welches im Juli 2024 veröffentlicht wurde. Zusätzlich verfügt das Modell über eine Kontextlänge von ca. 128000 Token. Es wurde für dialogbasierte Anwendungsfälle optimiert und unterstützt neben Englisch sieben weitere Sprachen, darunter Deutsch. Das Training umfasste circa 15 Billionen multilingualer Token aus öffentlich zugänglichen Quellen, wobei die Daten bis Dezember 2023 reichen. Das Modell wurde auf Basis von Supervised Fine-Tuning und RLHF angepasst [59].
- **Qwen 3 8B[11]:** Das von Alibaba entwickelte Qwen 3 8B ist ebenfalls ein quelloffenes Decoder-only Modell mit ca. 8,2 Milliarden Parametern, welches im April 2025 erschienen ist. Es unterstützt nativ eine Kontextlänge von ca. 32768 Token, die mittels „YaRN“-Skalierung auf bis zu 131072 Token erweitert werden kann. Zudem werden über 100 Sprachen und Dialekte, einschließlich Deutsch, unterstützt. Das Modell wurde für komplexe Schlussfolgerungsaufgaben („Reasoning“) sowie effiziente Dialoge konzipiert. Eine Besonderheit ist der nahtlose Wechsel zwischen einem „Thinking“-Modus, für anspruchsvolle Aufgaben, und einem „Non-Thinking“-Modus, für allgemeine Konversationen, wobei in dieser Arbeit ausschließlich der „Non-Thinking“-Modus zur besseren Vergleichbarkeit genutzt wurde. Das Pre-Training des Modells umfasste rund 36 Billionen Token, während im Post-Training das Modell über einen fünfstufigen Prozess weiter angepasst wurde [171].
- **Gemma 3 4B[12]:** Gemma 3 4B ist ein von Google entwickeltes, auch quelloffenes, Decoder-only Modell mit ca. 4 Milliarden Parametern, welches im März 2025 erschienen ist und welches über ein Kontextfenster von ca. 128000 Token verfügt. Dadurch, dass das Modell ca. 4 Milliarden Parameter besitzt, stellt es das kleinste Modell in der Versuchsgruppe dar, unterstützt jedoch über 140 Sprachen, darunter

[10] https://ai.meta.com/blog/meta-llama-3/ - Abgerufen am 16.06.2025
[11] https://qwenlm.github.io/blog/qwen3/ - Abgerufen am 16.06.2025
[12] https://deepmind.google/models/gemma/gemma-3/ - Abgerufen am 16.06.2025

auch Deutsch. Das Modell ist ein multimodales Modell, das Text- und Bildeingaben verarbeiten und Textausgaben generieren kann. Das Training des Modells umfasste 4 Billionen Token und wurde unter anderem durch Supervised Fine-Tuning und RLHF angepasst [150].

Die Auswahl dieser Modelle begründet sich zudem in der in Kapitel 3.4 beschriebenen und in dieser Arbeit genutzten Technologie zum Fine-Tuning der Modelle und der zu dieser Zeit unterstützten Modelle durch jene Technologie.
Zusätzlich zu den zu untersuchenden Modellen sind zu Vergleichszwecken ebenfalls SOTA-Modelle genutzt worden, deren Ergebnisse jedoch ohne die Nutzung des RAG-Systems oder Fine-Tuning erzielt wurden. Die ausgewählten Modelle bilden durch ihre Modelleigenschaften, wie Architektur und Parameteranzahl, sowie ihrer gemessenen Leistungen in verschiedenen Benchmarks einen guten Vergleich zu den kleineren Sprachmodellen. Zudem wurde bei der Auswahl darauf geachtet, ebenfalls ein Open-Source Modell als Vergleichsmodell heranzuziehen, damit eine bessere Vergleichbarkeit zwischen den zu untersuchenden kleineren Sprachmodellen gewährleistet ist. Außerdem war ein gutes Preis-Leistungsverhältnis sowie eine schnelle Inferenzzeit der Modelle von Bedeutung. Aus diesen Gründen wurden folgende SOTA-Modelle ausgewählt:

- **Gemini 2.5 Flash[13] und Pro Preview[14]:** Hierbei handelt es sich um zwei proprietäre Modelle von Google. Gemini 2.5 Flash ist für Geschwindigkeit und Effizienz bei gleichzeitig hoher Leistungsfähigkeit optimiert und bietet eine Kontextlänge von bis zu einer Million Token. Gemini 2.5 Pro ist das leistungsstärkste Modell der Familie für hochkomplexe Aufgaben und bietet standardmäßig ebenfalls eine Kontextlänge von einer Million Token. Beide Modelle sind multimodal, d.h. sie können Text, Code, Bilder, Audio und Video verarbeiten und generieren. Sie wurden aufgrund ihrer Spitzenleistung in Benchmarks und ihrer Fähigkeit, komplexe Anweisungen zu verstehen, ausgewählt. Insbesondere das „Flash“-Modell zielt auf ein sehr gutes Preis-Leistungsverhältnis und schnelle Inferenz ab, während das „Pro“-Modell durch Reasoning-Tokenausgaben im „Thinking“-Modus seine Performance verbessert.

- **Deepseek V3-0324[15]:** Dieses von DeepSeek AI entwickelte Modell ist ein Update ihres „V3“ Modells [37] vom Dezember 2024, wobei die Version „V3-0324“ im März

[13] https://deepmind.google/technologies/gemini/flash/ - Abgerufen am 16.06.2025
[14] https://deepmind.google/technologies/gemini/pro/ - Abgerufen am 16.06.2025
[15] https://api-docs.deepseek.com/news/news250325 - Abgerufen am 16.06.2025

2025 mit spezifischen Updates veröffentlicht wurde. Es handelt sich, im Gegensatz zu den Gemini-Modellen, um ein quelloffenes Sprachmodell, das auf einer „Mixture-of-Experts"-Architektur mit insgesamt 671 Milliarden Parametern basiert, von denen jedoch nur 37 Milliarden pro Token aktiviert werden. Das Modell unterstützt dabei eine Kontextlänge von ca. 128000 Token, während seine Architektur auf eine hohe Effizienz beim Training und der Inferenz abzielt. Das Modell wurde auf 14,8 Billionen Token vortrainiert und anschließend mittels Supervised Fine-Tuning und Reinforcement Learning optimiert. Die Version V3-0324 zeichnet sich insbesondere durch eine erhebliche Steigerung der Reasoning-Leistung aus und besitzt ebenfalls einen „Thinking"-Modus, welcher jedoch für diese Arbeit nicht genutzt wurde.

Zusammenfassend wurden für die Untersuchung der verschiedenen Optimierungsansätzen zur Generierung von Übersetzungen in Leichter Sprache aus Standardsprache insgesamt drei quelloffene, kleine Sprachmodelle von verschiedenen Anbietern genutzt und deren Ausgaben untereinander und mit drei SOTA-Modellen verglichen.

3.3 Erstellung des RAG-Systems für ICL

Nach der Datensatzaufbereitungsphase sowie der Modellauswahl wurde in diesem Schritt das RAG-System zur Nutzung der ICL-Technik implementiert. Dazu wurde zunächst die TM-Datenbank erstellt und darauffolgend die Datenbank innerhalb des RAG-Systems eingebettet. Damit wurde die Umsetzung der Grundidee nach Kopp et al. [83] angestrebt, die diese hybride Technik aus beispielbasierten und neuronalen Methoden insbesondere im öffentlichen Sektor zur Übersetzung von Standardsprache in Leichte Sprache bereits vorschlugen. Zusätzlich lässt sich die RAG-Architektur im Bereich des erweiterten RAGs einordnen, da hier explizit Pre- und Post-Retrievalprozesse implementiert werden.

3.3.1 Erstellung eines TMs

Zur Erstellung des TMs galt, wie auch bei der Modellauswahl, die Beachtung der Bedingungen innerhalb der ressourcenbeschränkten Umgebung des öffentlichen Sektors. Entsprechend wurde bei der Auswahl der Datenbank für das TM darauf geachtet, dass diese Lösung möglichst quelloffen und lokal lauffähig ist sowie die Möglichkeit anbietet, Vektoren zu speichern und abzurufen. Auf Basis dessen wurde sich für die Vektordatenbank

Qdrant[16] entschieden, da dieser nicht nur für den späteren Prozess der Vektorsuche innerhalb des RAG-Systems performant[17] ist und bereits Features zur Befüllung von Vektoren sowie Einbindung der Vektorsuche bietet, sondern zudem quelloffen und lokal ausführbar ist, durch die Nutzung einer SQLite-Datenbank[18].
SQLite als eingebettetes Datenbankverwaltungssystem zeichnet sich durch seine Portabilität und seinem kompakten Design aus. Durch umfangreiche Tests, einschließlich Simulationen von kritischen Systemausfällen, ist es äußerst zuverlässig. Darüber hinaus bietet SQLite Leistungsvorteile mit seiner Kapazität für hohe Transaktionsraten und effiziente Datenverarbeitung. Hauptaspekt für die Entscheidung ist die hohe Portabilität von SQLite, die darauf beruht, dass es als einzelne Datei vorliegt und auf jeder Plattform mit einem C-Compiler lauffähig ist. Die Datenbankdateien sind zudem binärkompatibel zwischen verschiedenen Architekturen, was den Datenaustausch vereinfacht. Eine Installation oder spezielle Konfiguration ist nicht erforderlich [53]. Entsprechend ist SQLite in einer lokalen Umgebung ideal einsetzbar und durch die hohe Portabilität kann die TM-Datenbank zwischen verschiedenen Systemen ausgetauscht und genutzt werden. Zur Befüllung des TMs mit entsprechenden Vektorrepräsentationen des präparierten Datensatzes konnte dabei auf die zusätzlichen Funktionalitäten von Qdrant zurückgegriffen werden.
Zunächst wurden bei der Erstellung des TMs die 5643 TUs aus dem Datensatz zur Befüllung vorbereitet. Dazu wurden die einzelnen Sätze in Standardsprache der TUs zur Indexierung bereitgestellt, während die Übersetzungseinheiten in Leichter Sprache als Metadaten abgespeichert werden sollten. Dies hatte den Grund, dass zur Suche geeigneter Übersetzungen innerhalb des RAG-Systems ein Abgleich zwischen den Sätzen der Standardsprache aus der Suchanfrage und dem TM erfolgen musste. Die entsprechenden Übersetzungsbeispiele der Sätze in Standardsprache aus den in den Metadaten befindlichen Sätzen in Leichter Sprache wurden anschließend für den ICL-Ansatz benötigt.
Zur Indexierung der Sätze in Standardsprache wurden zwei verschiedene Indexierungs- bzw. Embeddingmethoden angewandt. Zum Einen wurde für den semantischen Abgleich der Vektoren das Embeddingmodell `jina-embeddings-v2-base-de` verwendet, welches auch in der Datensatzaufbereitungsphase aus denselben Gründen genutzt wurde. Zum Anderen wurden ebenfalls Vektoren erstellt, um eine lexikalische Suche durchführen zu können. Dazu wurde das BM25-Modell[19] von Qdrant genutzt. Durch die Nutzung beider Vektortypen, „Dense“- und „Sparse“-Vektoren, konnte im später implementierten

[16] https://qdrant.tech/ - Abgerufen am 16.06.2025
[17] https://qdrant.tech/benchmarks/ - Abgerufen am 16.06.2025
[18] https://www.sqlite.org/index.html - Abgerufen am 16.06.2025
[19] https://huggingface.co/Qdrant/bm25 - Abgerufen am 16.06.2025

RAG-System eine hybride Suche umgesetzt werden, welches die Suchergebnisse verbesserte. Beide Modelle konnten mithilfe der ebenfalls in Qdrant befindlichen Bibliothek `FastEmbed`[20] für eine schnelle und akkurate Indexierung und Umwandlung in Vektorrepräsentationen von Sätzen zum späteren Abgleich eingebunden werden.

3.3.2 Implementation des RAG-Systems

Bei der Implementation des RAG-Systems musste zunächst eine geeignete Retrieval-Strategie ausgewählt werden. Die Literatur legt nahe, dass eine kombinierte Suche auf Basis von lexikalischer und semantischer Suche optimale Ergebnisse hinsichtlich Leistung und Effizienz liefern kann und gegenüber reiner semantischer oder lexikalischer Suche besser abschneidet [13, 32, 165]. Zusätzlich kann die Retrieval-Leistung durch ein Reranking-Modell, welches die gelieferten Suchergebnisse anhand ihrer Relevanz zur Suchanfrage neu ordnet, weiterhin verbessert werden [57, 152, 165].
Im Zuge dessen wurde eine Implementation umgesetzt, bei der jeweils einzelne Sätze in Standardsprache die Suchanfrage ergaben und anhand dieser Anfrage mehrere passende TUs in Leichter Sprache in einem ersten Schritt zurückgegeben und in einem zweiten Schritt anhand ihrer Relevanz zur Suchanfrage neu geordnet wurden. Diese zweiphasige Retrieval-Strategie entspricht auch der Idee nach Boyce [17]. In der Implementation wurden in der ersten Phase zehn Beispiele aus dem TM abgerufen und in der zweiten Phase fünf Beispiele durch das Reranking für den weiteren Prozess zurückgegeben.
Zur Umsetzung der hybriden Suche mit Reranking-Verfahren wurde die Klasse `HybridSearcher` implementiert, wobei Beispiele aus der Qdrant Dokumentation[21][22] herangezogen und für den Anwendungsfall angepasst wurden. Die Umsetzung der Klasse `HybridSearcher` kann unter A.8 eingesehen werden. Im Wesentlichen werden in der Klasse die einzelnen Funktionen für die Suche und das Reranking definiert. Neben der genannten Embedding-Modelle `jina-embeddings-v2-base-de` und `BM25` für die Suche, wurde für das Reranking das Cross-Encoder-Modell `jina-reranker-v2-base-multilingual`[23] verwendet, welches ebenfalls mit 278 Millionen Parametern leichtgewichtig, quelloffen und performant ist.
Objekte der Klasse `HybridSearcher` bilden dabei den ersten Teil des RAG-Systems,

[20] https://github.com/qdrant/fastembed - Abgerufen am 16.06.2025
[21] https://qdrant.tech/documentation/beginner-tutorials/hybrid-search-fastembed/ - Abgerufen am 16.06.2025
[22] https://qdrant.tech/documentation/fastembed/fastembed-rerankers/ - Abgerufen am 16.06.2025
[23] https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual - Abgerufen am 16.06.2025

dem Retriever. Zur Umsetzung des zweiten Teils des Systems, dem Reader bzw. Generator, wurde eine weitere Klasse `LLMTranslator`, siehe auch A.9 und Abbildung 3.5, implementiert.

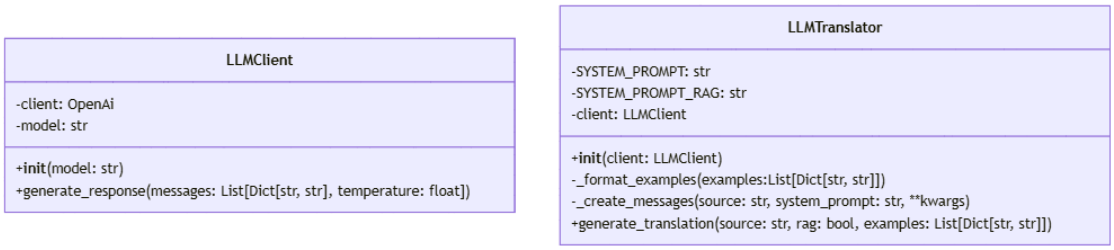

Abbildung 3.5: Klassendiagramm für die Klassen LLMClient und LLMTranslator

Um den Retriever- sowie den Reader- bzw. Generator-Teil miteinander zu verbinden, wurde des Weiteren die `Translator`-Klasse implementiert, der jeweils durch einen `QdrantClient` aus der `qdrant_client`-Bibliothek und den eigens implementierten `LLMClient` initialisiert wird. Bei der Initialisierung werden dabei die Objekte `HybridSearcher` und `LLMTranslator` für den Retriever respektive Reader bzw. Generator-Teil angelegt. Weiterhin kann bei der Nutzung der implementierten `translate`-Funktion entschieden werden, ob eine Suche von TUs gewünscht ist, sodass auch eine Übersetzung ohne Beispieldokumente ermöglicht wird. Dies war notwendig für die spätere Evaluation verschiedener Konfigurationseinstellungen. Entsprechend existieren auch je nach Konfiguration zwei verschiedene System-Prompts. Abschließend war es notwendig, dass einzelne Dokumente, die übersetzt werden sollten, jeweils in einzelne Sätze aufgesplittet werden konnten. Dafür wurde der regelbasierte Tokenizer SoMaJo[24] genutzt, der explizit auch für die Deutsche Sprache verwendet werden kann. Zur Evaluation wurde weiterhin geprüft, wie die einzelnen LLMs auch komplette Dokumente übersetzen konnten. Entsprechend gibt es in der `translate`-Funktion die Möglichkeit via `sentence_mode` auch komplette Dokumente statt einzelne Sätze zu übersetzen. Eine Darstellung des Klassendiagramms für den `Translator` mit `HybridSearcher` und `LLMTranslator` ist in Abbildung 3.6 einsehbar. Die Implementation der `Translator`-Klasse kann unter A.10 eingesehen werden. Zudem kann unter Abbildung 3.7 die Interaktion zwischen dem Nutzer, `Translator` sowie den dazugehörigen `HybridSearcher` und `LLMTranslator` in einem Sequenzdiagramm nachvollzogen werden. Das Sequenzdiagramm beschreibt dabei die vollständige Nutzung des Systems, bei Nutzung des RAG-Systems auf Satzbasis.

[24] https://github.com/tsproisl/SoMaJo - Abgerufen am 16.06.2025

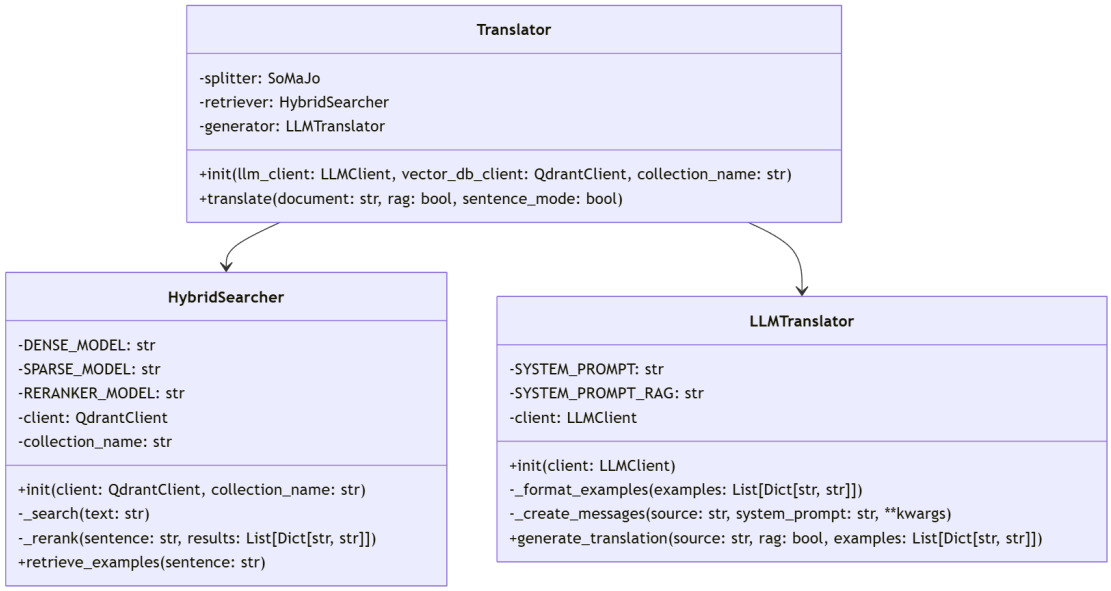

Abbildung 3.6: Klassendiagramm für die Klassen HybridSearcher, LLMTranslator und Translator

Durch Umsetzung der Retriever-Generator Architektur des RAG-Systems auf diese Weise, konnte für die spätere Evaluation die ausgewählten LLMs dynamisch getestet und verschiedene Konfigurationseinstellungen in Betracht gezogen werden.

3.4 Fine-Tuning der Modelle

Neben der Implementation des RAG-Systems wurde in dieser Arbeit des Weiteren untersucht, inwiefern ein Fine-Tuning von LLMs sich zu diesem Anwendungsfall und in Kombination mit dem RAG-System auf die Übersetzungsqualität auswirkt. Dazu wurden die ausgewählten Modelle im weiteren Prozess jeweils durch die Fine-Tuning Technik LoRA [70] weiter angepasst. Insbesondere der Einsatz der Technik QLoRA ermöglichte eine Anpassung der LLMs in einer ressourcenschonenden und effizienten Weise durch die Quantisierung der vortrainierten Modelle auf 4-Bit und weiterer Speicheroptimierungen Im Gegensatz zu LoRA reduziert QLoRA nicht nur die Anzahl der zu trainierenden Parameter, sondern auch den Speicherbedarf des Basismodells selbst, was eine noch höhere Effizienz beim Fine-Tuning ermöglicht [39].
Für das Fine-Tuning der Modelle selbst wurde Unsloth.ai[25] genutzt [62]. Bei Unsloth wird

[25] https://unsloth.ai - Abgerufen am 16.06.2025

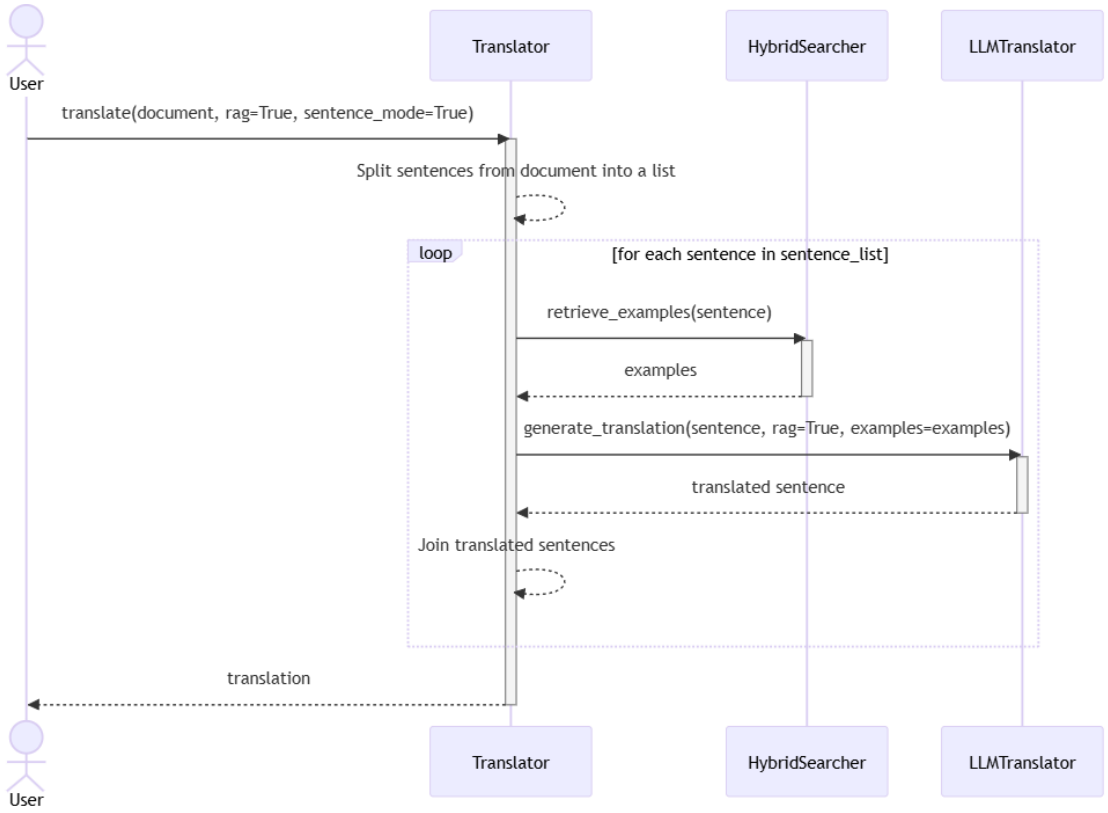

Abbildung 3.7: Sequenzdiagramm der Interaktion zwischen Nutzer, Translator, HybridSearcher und LLMTranslator

Triton [154] für die Implementierung von Backpropagation in LoRA-Trainingsprozessen eingesetzt. Die Verwendung von Triton ermöglicht es, die Anzahl der Fließkommaoperationen („FLOPs“) während des Gradientenabstiegs zu minimieren, was sich positiv auf die Trainingsgeschwindigkeit auswirkt [179]. Unsloth gibt Entwicklern zudem die Möglichkeit des Fine-Tunings via QLoRA mit verschiedene unterstützten Open-Source Modellen über Google Colab[26] an, einem von Google gehostetem Jupyter Notebook Service. Unsloth bietet weiterhin bereits einige vorgefertigte Google Colab Notebooks[27], die als Vorlage verwendet und entsprechend den Anforderungen angepasst worden sind. Zusätzlich bietet Unsloth über Huggingface vorerstellte 4-bit quantisierte Modelle durch Nutzung von bitsandbytes[28], welches k-Bit-Quantisierung in PyTorch ermöglicht [71], wodurch der Speicherbedarf während des Trainings und der Inferenz weiter reduziert wird. Dadurch

[26] https://colab.research.google.com/ - Abgerufen am 16.06.2025
[27] https://docs.unsloth.ai/get-started/unsloth-notebooks - Abgerufen am 16.06.2025
[28] https://github.com/bitsandbytes-foundation/bitsandbyte - Abgerufen am 16.06.2025

lässt sich auch ein Fine-Tuning mit QLoRA bewerkstelligen [12].
Dadurch, dass die ausgewählten Modelle `Llama 3.1 8B`, `Qwen 3 8B` und `Gemma 3 4B` ebenfalls als quantisierte Modelle über Huggingface durch Unsloth.ai verfügbar sind und aufgrund der hohen Effizienz und ressourcenschonenden Art, um ein Fine-Tuning der Modelle durchzuführen, wurde sich in dieser Arbeit aus diesem Grund für Unsloth.ai entschieden.
Der ShareGPT-Datensatz mit den jeweils 500 TUs wurden für die jeweiligen Trainings-Notebooks über Google Drive zur Verfügung gestellt und für den weiteren Trainingsprozess bei der Ausführung des Notebooks noch weiter angepasst, indem dem Datensatz ein `text`-Element hinzugefügt wurde, das in der späteren Konfiguration notwendig war.
Für das Herunterladen der ausgewählten Modelle und der jeweiligen Tokenizer, wurden in einem ersten Schritt die entsprechenden Einstellungen vorgenommen. Tabelle 3.2 zeigt eine Beispielkonfiguration für das 4-bit quantisierte Llama 3.1-8B-Instruct Modell. Die maximale Sequenzlänge für das Training wurde auf 2048 Token gesetzt. Außerdem wurden die Parameter `load_in_4bit` für die 4-bit quantisierte Version sowie `dtype` mit None initialisiert, um den Datentyp automatisch zu erkennen.

Parameter	**Wert**
`max_seq_length`	`2048`
`load_in_4bit`	`True`
`dtype`	`None`

Tabelle 3.2: Beispielkonfiguration beim Laden der Modelle

Im nächsten Schritt mussten die QLoRA-Konfiguration initialisiert werden. Die Standardeinstellungen aus den Notebookvorlagen wurden in diesem Fall übernommen. Eine Beispielkonfiguration ist in Tabelle 3.3 zu sehen. Eine genaue Auflistung der Parameter und deren Auswirkungen lässt sich unter der Unsloth-Dokumentation einsehen[29]. Zusammengefasst wurde ein kleinerer Rang `r=8` und ein dazu passendes `lora_alpha=8` für eine schnelle Trainingsiterationen und zur Vermeidung von Overfitting bei gleichzeitig guter Genauigkeit ausgewählt. Unsloth optimiert für `lora_dropout=0` und `bias="none"`, was ebenfalls zu schnellerem Training führt und als Standard für viele Fälle empfohlen wird. Erweiterte Funktionen wie `use_rslora` und `loftq_config` blieben deaktiviert, da die Standard-QLoRA-Konfiguration oft bereits sehr gute Ergebnisse liefert.

[29] https://docs.unsloth.ai/get-started/beginner-start-here/lora-parameters-encyclopedia - Abgerufen am 16.06.2025

Parameter	Wert
finetune_vision_layers	False
finetune_language_layers	True
finetune_attention_modules	True
finetune_mlp_modules	True
r	8
lora_alpha	8
lora_dropout	0
bias	none
random_state	3407
use_rslora	False
loftq_config	None

Tabelle 3.3: Beispielkonfiguration für den QLoRA-Layer

Im letzten Schritt vor dem Trainingsbeginn, musste der Trainer initialisiert werden. Dieser basiert auf dem `SFTTrainer`[30], der weitere Trainingsargumente über das Modul `SFTConfig` erhält. Auch in diesem Fall wurde weitestgehend die Standardeinstellungen der Vorlage übernommen, jedoch minimal angepasst. So wurden die Parameter `per_device_train_batch_size` auf 2 und `gradient_accumulation_steps` auf 4 gesetzt. Zusätzliche ist die Anzahl der Trainingsepochen auf 2 festgelegt worden. Die Wahl von insgesamt 2 Epochen wurde getroffen, um einen schnelleren und kostensparenden Trainingsdurchlauf zu ermöglichen, ohne ein Overfitting zu verursachen und da eine Epochenzahl über 3 „nicht optimal“[31] sei. Das Setzen der Parameter `per_device_train_batch_size` und `gradient_accumulation_steps` sorgte für eine Minimierung der Speicheranforderungen auf der GPU, während gleichzeitig eine akzeptable Batchsize für den Gradientenabstieg beibehalten wurde. Dies ergab eine effektive Batchsize von 8 ($2 * 4$), während der tatsächliche Speicherverbrauch durch die kleinere Batchsize auf der GPU reduziert wurde. Abschließend definiert die `warmup_ratio` den Anteil der initialen Trainingsschritte, in denen die Lernrate zur Stabilisierung langsam von 0 auf den Zielwert ansteigt, bevor das eigentliche Training mit der eingestellten Lernrate beginnt. Eine genaue Beschreibung der einzelnen Parameter für `SFTTrainer`[32] und

[30] https://huggingface.co/docs/trl/sft_trainer - Abgerufen am 16.06.2025
[31] https://docs.unsloth.ai/get-started/beginner-start-here/lora-parameters-encyclopedia - Abgerufen am 16.06.2025
[32] https://huggingface.co/docs/trl/main/trainer#trl.SFTTrainer - Abgerufen am 16.06.2025

`SFTConfig`[33] lässt sich in den einzelnen Dokumentationen nachlesen. Tabelle 3.4 zeigt die vorgenommenen Einstellungen für den Trainer.

Parameter	**Wert**
dataset_text_field	text
per_device_train_batch_size	2
gradient_accumulation_steps	4
num_train_epochs	2
learning_rate	2e-4
logging_steps	1
optim	adamw_8bit
weight_decay	0.01
lr_scheduler_type	linear
warmup_ratio	0.06
seed	3407
report_to	none
save_strategy	epoch

Tabelle 3.4: Beispielkonfiguration für den SFTTrainer durch SFTConfig

Nachdem das Training für jedes der ausgewählten Modelle abgeschlossen war, konnte eine Evaluation der einzelnen Modelle durchgeführt werden. Für die Evaluation der Fine-Tuning Modelle, wurde die vorhandene Implementation in ein Google Colab Notebook überführt und die native Inferenzlösung von Unsloth[34] genutzt, die keine Codeänderungen der Implementation erforderte. Ein komplettes Notebook für das Fine-Tuning des `Llama 3.1 8B`-Modells lässt sich unter A.11 einsehen. Dieses Notebook ließ sich analog zu den anderen Modellen durch Anpassung der geladenen Modelle entsprechend auf die selbe Weise für das Fine-Tuning nutzen.

[33] https://huggingface.co/docs/trl/main/en/sft_trainer#trl.SFTConfig - Abgerufen am 16.06.2025
[34] https://docs.unsloth.ai/basics/running-and-saving-models/inference - Abgerufen am 16.06.2025

3.5 Evaluation

3.5.1 Versuchsaufbau

Für die Evaluation der ausgewählten Modelle wurden verschiedene Konfigurationen ausgewählt, um ein umfassendes Bild verschiedener Übersetzungslösungen unter Nutzung verschiedener Techniken und Vergleichsmodellen zu erhalten. Dabei wurden die Hyperparameter der Modelle zur Inferenz, wie z.B. Temperature, auf die Standardwerte von OpenRouter[35] angepasst, um eine Vergleichbarkeit der Modelle zu gewährleisten. Zudem sind vier verschiedene Metriken genutzt worden, um die generierten Texte zu untersuchen. Diese Untersuchung fokussierte sich auf die drei zentralen Aspekte Einfachheit, Grammatikalität und Semantik. Zur Bewertung der Einfachheit wurden primär `Flesch-Reading-Ease` [51] und die spezifischen Vereinfachungsaspekte von `SARI` [170] herangezogen. Die Grammatikalität wurde indirekt durch `BLEU` [118] und `SARI` erfasst, indem ein Abgleich mit Referenztexten erfolgte. Für die Überprüfung der Semantik bzw. des Bedeutungserhalts kam insbesondere der `BERTScore` [177] zum Einsatz, wobei hier auch `BLEU` unterstützend wirkt. Die verschiedenen Konfigurationen umfassten folgende Aspekte:

- **Referenz**: Zur Ermittlung des Deltas der verschiedenen generierten Texte zur Originalübersetzung, wurden die ausgewählten Metriken auf diese in einem ersten Schritt ermittelt.
- **SLMs - Dokument:** In einem ersten Versuch wurden die Übersetzungen ohne die Anwendung verschiedener Techniken und auf Dokumentenbasis umgesetzt. Dabei wurde das zu übersetzende Dokument jeweils im Ganzen über den System-Prompt an die ausgewählten SLMs zur Übersetzung gegeben und daraufhin die verschiedenen Metriken ermittelt.
- **SOTA - Dokument:** Zum Vergleich mit der `SLMs - Dokument` Konfiguration wurde des Weiteren die ausgewählten SOTA-Modelle ebenfalls über einen System-Prompt auf die gleiche Weise dazu angewiesen, die Dokumente im Ganzen zu übersetzen. Anschließend wurden die ausgewählten Metriken ermittelt.
- **SLMs - Satz:** In einem weiteren Versuch wurden die einzelnen Dokumente des Evaluationsdatensatzes jeweils Satzweise von den ausgewählten Modellen übersetzt. In

[35] https://openrouter.ai/docs/api-reference/parameters - Abgerufen am 16.06.2025

diesem ersten Schritt wurde die Übersetzungsqualität der ausgewählten Basismodelle ohne Verwendung einzelner Optimierungsansätze untersucht. Dabei wurden die einzelnen Dokumente in einzelne Sätze gesplittet und jeweils über den Einsatz von System-Prompts mit den Modellen übersetzt. Die Übersetzungen der Sätze wurden anschließend zusammengeführt und die verschiedenen Metriken ermittelt.

- **SOTA - Satz:** Um ebenfalls in dieser Konfiguration einen Vergleich mit den SLMs zu erhalten, wurde auf die selbe Weise wie in der `SLMs - Satz` Konfiguration die einzelnen Sätze der Dokumenten über die SOTA-Modelle übersetzt und die Metriken anschließend ermittelt. Für die Untersuchung der verschiedenen Optimierungsansätze wurde im weiteren Verlauf auf die Verwendung der SOTA-Modelle verzichtet, da diese Ansätze nur in Bezug auf SLMs untersucht werden sollten.

- **SLMs - Satz - Fine-Tuning:** In dieser Konfiguration wurden die generierten Texte der ausgewählten SLMs untersucht, die über das Modell mit Fine-Tuning generiert worden sind. Da das Fine-Tuning der SLMs darauf abzielte einzelne Sätze zu übersetzen, wurden die Dokumente aus dem Evaluationsdatensatz analog zur `SLMs - Satz` Konfiguration übersetzt und die Metriken des generierten Dokuments ermittelt.

- **SLMs - Satz - ICL:** Neben dem Fine-Tuning wurden die generierten Übersetzungen der ausgewählten SLMs ebenso anhand der Nutzung des RAG-Systems erhoben und untersucht, inwiefern ICL die Generierung beeinflusst. Da das TM lediglich TUs auf Satzbasis enthält, wurde die Konfiguration ebenfalls ähnlich zur `SLMs - Satz` Konfiguration ausgewählt und anschließenden die Metriken ermittelt.

- **SLMs - Satz - Fine-Tuning und ICL:** Abschließend wurde überprüft welchen Einfluss die Nutzung beider Techniken, Fine-Tuning und ICL, auf die Generierung der Texte von SLMs hat. Analog zu den vorherigen genannten Experimenten, wurde die gleiche Konfiguration genutzt und die Metriken der generierten Dokumente ermittelt.

Zur Ermittlung der Werte für die einzelnen Metriken wurde einerseits die Bibliothek `evaluate`[36] sowie `textstat`[37] verwendet.
Die `evaluate`-Bibliothek bietet Implementationen von verschiedenen Evaluationsmetriken für Machine Learning Modelle und Datensätzen an, darunter auch die in dieser

[36] https://github.com/huggingface/evaluate - Abgerufen am 16.06.2025
[37] https://github.com/textstat/textstat - Abgerufen am 16.06.2025

Arbeit genutzten BLEU, SARI und BERTScore. Zur Ermittlung des BERTScore wurde bei der Auswahl der deutschen Sprache in der Konfiguration das Standardmodell bert-base-multilingual-cased genutzt. textstat bietet auf der anderen Seite die Möglichkeit Lesbarkeitsmetriken, wie die in dieser Arbeit genutzten Flesch-Reading-Ease-Metrik, zu ermitteln. Nach der Generierung der Übersetzungen mit allen genannten Konfigurationen wurden im Anschluss die Ermittlung der einzelnen Metriken für jede Übersetzung durchgeführt und über alle Konfigurationen jeweils der Durchschnittswert sowie der Medianwert ermittelt. Mit dem Medianwert sollte sichergestellt werden, dass Ausreißerwerte bei den Metriken, z.B. Minuswerte bei der Flesch-Reading-Ease-Metrik, nicht ins Gewicht fallen. Vollständigkeitshalber werden deswegen beide Werte jeweils aufgeführt.

3.5.2 Ergebnisse

Die Ergebnisse der durchgeführten Versuche werden in den folgenden Tabellen detailliert dargestellt. Tabelle 3.5 fasst die Durchschnitts- und Medianwerte der vier ausgewählten Evaluationsmetriken für alle untersuchten Konfigurationen zusammen. Tabelle 3.6 zeigt die prozentuale relative Verbesserung dieser Metriken - ausgenommen des SARI-Wertes, da dieser nur absolut vorliegt - im Vergleich zur Referenzkonfiguration.
Die Referenzkonfiguration weist für Flesch-Reading-Ease einen Durchschnittswert von 70.536 und einen Median von 72.220 auf. Der BLEU-Score der Referenz liegt bei 0.424 bzw. 0.430 und der BERTScore bei 0.693 bzw. 0.688. Für die SARI-Metrik ist bei der Referenz kein Wert angegeben, da diese für die Referenzkonfiguration nicht anwendbar ist. Bei allen Metriken in der Tabelle 3.5 signalisieren höhere Werte bessere Ergebnisse, und die jeweils höchsten Werte pro Spalte sind fett markiert.

Für Flesch-Reading-Ease erzielt die Konfiguration SOTA – Satz mit einem Durchschnittswert von 77.312 den höchsten Wert, jedoch erreichte die Konfiguration SLM – Satz – Fine-Tuning den höheren Flesch-Reading-Ease Medianwert von 79.143. Die niedrigsten Flesch-Reading-Ease-Werte liefert die Konfiguration SLM – Dokument mit einem Durchschnittswert von 66.171 bzw. einem Median von 67.237, was unter dem Referenzwert liegt. Im Hinblick auf den BLEU-Score erreichte die Konfiguration SLM – Satz den Spitzenwert im Durchschnitt von 0.577. Im Median schnitt jedoch die Konfiguration SLM – Satz – ICL mit 0.596 besser ab. Die Konfiguration SLM – Satz – ICL folgt aber auch im Durchschnittswert direkt dahinter mit dem Wert von 0.573, also einem Unterschied von 0.004. Den niedrigsten BLEU-Score unter den getesteten

Konfiguration/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
Referenz	70,536	72,220	0,424	0,430	0,693	0,688	—	—
SLM - Dokument	66,171	67,237	0,552	0,561	**0.791**	**0.792**	42,394	42,281
SOTA - Dokument	74,272	75,277	0,495	0,498	0,734	0,731	**48.732**	48,028
SLM - Satz	68,308	69,115	**0.577**	0,582	0,786	0,787	42,537	42,239
SOTA - Satz	**77.312**	77,565	0,514	0,515	0,731	0,729	48,608	**48.489**
SLM - Satz - FT	76,780	**79.143**	0,566	0,572	0,755	0,756	43,786	43,256
SLM - Satz - ICL	68,790	72,137	0,573	**0.596**	0,767	0,770	44,788	44,213
SLM - Satz - FT+ICL	75,352	73,508	0,565	0,574	0,747	0,746	45,382	44,560

Anmerkung: FRE: Flesch-Reading-Ease; FT: Fine-Tuning; ICL: In-Context Learning. Diese Tabelle zeigt Durchschnitts- (Avg) und Medianwerte (Med), die als Durchschnitt der entsprechenden Avg/Med-Werte aus den detaillierten Modelltabellen berechnet wurden. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken, auch gekennzeichnet durch ↑; „—“ bedeutet nicht anwendbar; Die höchsten Werte je Spalte sind fett markiert.

Tabelle 3.5: Vergleich des Durchschnitts (Avg) und des Medians (Med) der Metriken aller Konfigurationen

Konfigurationen weist der Referenzwert mit einem Durchschnittswert von 0.424 bzw. einem Medianwert von 0.430, gefolgt von `SOTA - Dokument` mit 0.495 bzw. 0.498. Beim BERTScore zeigt die Konfiguration `SLM - Dokument` die besten Ergebnisse mit einem Durchschnittswert von 0.791 und einem Median von 0.792. Auch die Konfiguration `SLM - Satz` erzielt ebenfalls hohe Durchschnitts- und Medianwerte mit 0.786 bzw. 0.787. Die niedrigsten BERTScore-Werte unter den Konfigurationen werden von der Referenz erreicht mit 0.693 bzw. 0.688, gefolgt von der Konfiguration `SOTA - Satz` mit 0.731 bzw. 0.729. Für die SARI-Metrik erzielt `SOTA - Dokument` den höchsten Durchschnittswert von 48.732, während die Konfiguration `SOTA - Satz` den höchsten Medianwert von 48.489 aufweist. Die Konfiguration `SLM - Dokument` und `SLM - Satz` zeigen die niedrigsten SARI-Werte mit 42.394 bzw. 42.281 und 42.537 bzw. 42.239. Die SLMs-Konfigurationen mit den Optimierungstechniken erreichen durchweg höhere SARI-Werte, wobei die Konfiguration `SLM - Satz - Fine-Tuning+ICL` mit dem Durchschnittswert von 45.382 und dem Medianwert von 44.560 die besten Werte unter diesen Konfigurationen liefert.

Um einen besseren Überblick über die relativen Verbesserung hinsichtlich der Referenzkonfiguration zu erhalten, wird in der Tabelle 3.6 die prozentuale Veränderung der Metriken quantifiziert. Der SARI-Wert wird dabei nicht aufgeführt, da er lediglich als Absolutwert vorliegt. Bezüglich der relativen Verbesserung des Flesch-Reading-Ease-Wertes zeigt die Konfiguration `SOTA - Satz` die stärkste Zunahme mit +9.61 % im Durch-

schnitt. Im Median schnitt dagegen die Konfiguration `SLM - Satz - Fine-Tuning` mit +9.59 % besser ab. Einige Konfigurationen, wie `SLM - Dokument`, `SLM - Satz` und `SLM - Satz - ICL`, weisen jedoch auch eine Verschlechterung der Lesbarkeit im Vergleich zur Referenz in Höhe von -6.19 % bzw. -6.90 % und -3.16 % bzw. -4.30 % sowie -2.48 % bzw. -0.11 % auf. Bei der relativen Verbesserung des BLEU-Wertes zeigen alle Konfigurationen signifikant positive Werte. Die Konfiguration `SLM - Satz` erzielt dabei die höchste Steigerung mit +36.08 % im Durchschnitt, während im Median die Konfiguration `SLM - Satz - ICL` mit einer Steigerung von +38.60 % das beste Ergebnis erzielte. Die geringste, aber immer noch positive, Verbesserung zeigt dabei die Konfiguration `SOTA - Dokument` mit einem Durchschnittswert von +16.75 % und einem Medianwert von +15.81 %.

Konfiguration/Metrik	FRE ↑		BLEU ↑		BERTScore ↑	
	Avg	Med	Avg	Med	Avg	Med
SLM - Dokument	−6,19	−6,90	30,19	30,47	**14.14**	**15.12**
SOTA - Dokument	5,30	4,23	16,75	15,81	5,92	6,25
SLM - Satz	−3,16	−4,30	**36.08**	35,35	13,42	14,39
SOTA - Satz	**9.61**	7,40	21,23	19,77	5,48	5,96
SLM - Satz - FT	8,85	**9.59**	33,49	33,02	8,95	9,88
SLM - Satz - ICL	−2,48	−0,11	35,14	**38.60**	10,68	11,92
SLM - Satz - FT+ICL	6,83	1,78	33,25	33,49	7,79	8,43

Anmerkung: FRE: Flesch-Reading-Ease; FT: Fine-Tuning; ICL: In-Context Learning. Relative Verbesserungen basierend auf den Werten aus Tabelle 3.5. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken, auch gekennzeichnet durch ↑. Die höchsten Werte je Spalte sind fett markiert; Da der SARI-Wert absolut vorliegt, wird er in dieser Tabelle nicht dargestellt.

Tabelle 3.6: Relative Verbesserung in Prozent der Metriken im Durchschnitt (Avg) und des Median (Med) gegenüber der Referenz

Abschließend, hinsichtlich der relativen Verbesserung des BERTScores, erzielte die Konfiguration `SLM - Dokument` die größten Zuwächse mit +14.14 % im Durchschnitt und +15.12 % im Median. Die Konfiguration `SLM - Satz` zeigt ebenfalls deutliche Verbesserungen mit +13.42 % respektive +14.39 %. Die geringsten relativen Verbesserungen im BERTScore werden für die Konfiguration `SOTA - Satz` verzeichnet mit +5.48 % bzw. +5.96 %.

Die dargestellten Werte in den Tabellen 3.5 und 3.6 zeigen aggregierte Werte aller Modelle in ihrer jeweiligen Konfiguration. Eine detailliertere Aufschlüsselung aller Werte in

den Konfigurationen lässt sich unter A.12 finden. Die Medianwerte der einzelnen Konfigurationen in A.12 werden dabei für die Tabelle 3.5 als Durchschnittswert aggregiert.

4 Diskussion

Im Folgenden sollen im Kontext der eingangs formulierten Forschungsfragen die Evaluationsergebnisse interpretiert und diskutiert werden. Zunächst erfolgt eine Zusammenfassung und Interpretation der Ergebnisse in Bezug auf jede Forschungsfrage, wobei die Leistungen der Modelle insgesamt, aber auch die spezifischen einzelnen Modelle betrachtet werden. Abschließend werden die Resultate in den breiteren Forschungskontext eingeordnet und Limitationen der Arbeit werden reflektiert.

4.1 Interpretation der Ergebnisse im Kontext der Forschungsfragen

4.1.1 Fähigkeit von LLMs zur Erstellung von Leichter Sprache

Mit der ersten Forschungsfrage sollte untersucht werden, inwieweit LLMs in der Lage sind, automatisiert Texte in Leichter Sprache im behördlichen Kontext zu erstellen, die den Kriterien hinsichtlich Einfachheit, Grammatikalität und Semantik entsprechen. Die Basisleistungen, also ohne die Anwendung von Optimierungsstrategien, wurden anhand der Konfigurationen `SLM - Dokument`, `SOTA - Dokument`, `SLM - Satz` und `SOTA - Satz`, siehe auch die zusammenfassenden Tabellen 3.5 und 3.6 sowie die detaillierteren Tabellen A.3, A.4, A.5 und A.6, bewertet.

Einfachheit

Die Evaluation der Einfachheit, primär erfasst durch den Flesch-Reading-Ease-Wert und sekundär mit dem SARI-Wert, offenbarte einige Leistungsunterschiede. Insbesondere SOTA-Modelle zeigten sowohl bei dokumenten- als auch satzweiser Verarbeitung Flesch-Reading-Ease-Werte, die jene der menschlichen Referenz signifikant übertrafen. Dies legt

eine hohe Kapazität dieser Modelle zur Generierung sprachlich gut zugänglicher Texte nahe. Bei den SLMs konnte insbesondere Gemma 3 4B überzeugen und erreichte, vor allem bei satzweiser Verarbeitung, ebenfalls Werte oberhalb der menschlichen Referenz. Andere SLMs wie Llama 3.1 8B und Qwen3 8B erreichten in den Basiskonfigurationen hingegen nicht durchgängig das Einfachheitsniveau der Referenz. Aggregiert betrachtet bestätigte sich, dass `SOTA` Konfigurationen im Durchschnitt die höchsten Flesch-Reading-Ease-Werte erzielten, was auf das potenziell umfangreichere Wissen durch eine höhere Parameteranzahl und komplexeren Architekturen dieser Modelle zurückgeführt werden könnte. Die SLMs Konfigurationen lagen im Mittel unter dem Referenzwert, was den Bedarf an Optimierungsstrategien für diese Modellklasse unterstreicht, um das gewünschte Einfachheitsniveau zuverlässig zu erreichen.
Hinsichtlich der SARI-Werte ist festzustellen, dass vor allem die SOTA Konfigurationen auf Satz- und auf Dokumentebene am besten abschneiden und dadurch eine höhere Einfachheit der Texte suggerieren, vor allem bei gleichzeitiger Beachtung des Flesch-Reading-Ease-Wertes. Konfigurationen mit Fine-Tuning und in Kombination mit ICL zeigen jedoch auch, dass SLMs bei entsprechender Optimierung ebenfalls höhere SARI-Werte erzielen können als ihre Basiskonfigurationen. Abschließend können SLMs bereits in ihren Basisvarianten Texte generieren, die sich dem Einfachheitsniveau menschlicher Übersetzungen annähern und durch die genutzten Optimierungsstrategien sogar bessere Ergebnisse erzielen. Die satzweise Verarbeitung erwies sich tendenziell als vorteilhafter für die Steigerung der Einfachheit, möglicherweise, da sie eine fokussiertere Anwendung von Vereinfachungsoperationen auf kleinere Texteinheiten erlaubt.

Grammatikalität

Die Bewertung der Grammatikalität erfolgte über die referenzbasierten Metriken, allen voran BLEU und zusätzlich mit SARI. Wie dargelegt, korreliert die BLEU-Metrik zwar mit Aspekten der Grammatikalität und Semantik, jedoch weniger stark mit der Einfachheit und die SARI-Metrik korreliert stärker mit Einfachheit und Grammatikalität [58, 98, 170]. Überraschenderweise erzielte das Modell Qwen3 8B die höchsten BLEU-Werte und übertraf damit nicht nur andere SLMs, sondern auch die getesteten SOTA-Modelle. Die hohen BLEU-Werte aller Modellkonfigurationen, die jene der menschlichen Referenz deutlich übertrafen, deuten auf eine hohe lexikalische und syntaktische Ähnlichkeit der maschinell generierten Texte zur Referenzübersetzung hin. Insbesondere die SLMs Konfigurationen zeigten hier im Schnitt die stärksten Ergebnisse. Ein differenzierteres Bild

ergab sich bei der SARI-Metrik. Hier lagen die SOTA-Modelle tendenziell vorne, was darauf hindeutet, dass diese Modelle effektiver in den spezifischen Transformationen sind, die für eine gute Vereinfachung notwendig sind und über reine N-Gramm-Übereinstimmung hinausgehen. Die Ergebnisse zur Grammatikalität sind somit vielschichtig. Einerseits suggerieren die hohen BLEU-Werte eine gute Fähigkeit der Modelle, syntaktisch plausible und lexikalisch der Referenz ähnelnde Texte zu generieren. Andererseits deuten die SARI-Ergebnisse darauf hin, dass SOTA-Modelle überlegen sein könnten, wenn es um die adäquate Durchführung von Vereinfachungsoperationen geht, die für die Textqualität in der Leichten Sprache entscheidend sein können.

Semantik

Die semantische Äquivalenz, gemessen primär durch den BERTScore-Wert, zeigte ebenfalls bemerkenswerte Ergebnisse. Analog zu den BLEU-Ergebnissen stach Qwen3 8B unter allen Einzelmodellen hervor und erzielte die höchsten BERTScore-Werte, was auf eine hohe Beibehaltung der ursprünglichen Bedeutung hindeutet. Generell erreichten alle Konfigurationen von LLMs BERTScore-Werte, die jene der menschlichen Referenz deutlich übertrafen. Dies legt nahe, dass die Modelle die Bedeutung des Ausgangstextes sehr gut konservieren. Überraschenderweise erzielten auch hier die SLMs Konfigurationen im Durchschnitt höhere Werte. Ein besonders positives Ergebnis ist die durchweg hohe semantische Äquivalenz der von den LLMs generierten Texte. Die Beobachtung, dass insbesondere SLMs hier überzeugen, könnte darauf hindeuten, dass diese Modelle, möglicherweise aufgrund ihrer Architektur oder Trainingsdaten eine sehr enge Orientierung am Ausgangstext priorisieren, was jedoch auch abhängig von den gewählten Hyperparametern, wie z.B. der Temperature, sein könnte. Die höheren Werte gegenüber der menschlichen Referenz im BERTScore könnte darauf zurückzuführen sein, dass die Modelle sich strikter am Quelltext orientieren, während menschliche Übersetzer möglicherweise interpretative Freiheiten nutzen, die zu leichten semantischen Differenzen führen können, welche durch den BERTScore als geringere Ähnlichkeit bewertet wird. Zudem fällt unter Betrachtung der Flesch-Reading-Ease-Werte auf, dass ein höherer BERTScore im Durchschnitt mit einer Abnahme des Flesch-Reading-Ease-Wertes korreliert. Dies kann bedeuten, dass eine höhere semantische Ähnlichkeit zweier Texte die Einfachheit dieser beeinträchtigt und umgekehrt, was die Behauptungen der negativen Korrelation von Einfachheit und Semantik von Schwarzer und Kauchak [135] bestätigen würde.

Zusammenfassend lässt sich die erste Forschungsfrage differenziert beantworten. LLMs demonstrieren eine grundsätzliche Fähigkeit zur automatisierten Erstellung von Texten in Leichter Sprache im behördlichen Kontext, die den untersuchten Kriterien entsprechen. Es zeigen sich jedoch deutliche Unterschiede sowohl zwischen den Modellklassen als auch den einzelnen Metriken, was die Komplexität der Evaluation von Textvereinfachung unterstreicht. Während SOTA-Modelle oft eine höhere sprachliche Einfachheit und bessere Performanz bei spezifischen Vereinfachungsoperationen zeigen, können SLMs bei der Grammatikalität und insbesondere bei der Erhaltung der Semantik überzeugen. Das Modell Gemma 3 4B kristallisiert sich dabei als Modell heraus, das einen guten Kompromiss über verschiedene Bewertungsaspekte hinweg darstellt und aufgrund seiner geringeren Größe am effizientesten Texte in Leichter Sprache generieren kann. Die Wahl des optimalen Modells hängt jedoch stark von den spezifischen Anforderungen und der Gewichtung der einzelnen Qualitätskriterien ab.

4.1.2 Wirksamkeit der ausgewählten Optimierungsstrategien

Die zweite Forschungsfrage untersuchte, inwieweit ausgewählte spezifische Optimierungsstrategien die Performanz von SLMs bei der automatischen Erstellung von Texten in Leichter Sprache verbessern können. Im Fokus standen dabei die Ansätze des Fine-Tunings, des ICL mittels RAG sowie deren kombinierter Einsatz, deren Ergebnisse in den Tabellen 3.5 und 3.6 sowie in den detaillierteren Tabellen A.7, A.8 und A.9 dargestellt sind.

Fine-Tuning

Fine-Tuning erwies sich als eine äußerst effektive Strategie, um insbesondere die sprachliche Einfachheit der SLMs signifikant zu verbessern. Durch das Training auf dem spezifischen Korpus für Leichte Sprache, konnten alle untersuchten SLMs ihre Flesch-Reading-Ease-Werte deutlich steigern. Modelle wie Qwen3 8B mit Fine-Tuning und Gemma 3 4B mit Fine-Tuning erreichten hierdurch Werte, die mit den SOTA-Modellen konkurrieren oder diese im Median sogar übertreffen. Dies unterstreicht die Fähigkeit des Fine-Tunings der Spezialisierung der Modelle auf die stilistischen Eigenschaften der Leichten Sprache, wie z.B. kürzere Sätze und einfacheren Wortschatz. Auch hinsichtlich der SARI-Metrik zeigten sich positive Effekte. Insbesondere Qwen3 8B mit Fine-Tuning profitierte stark und erzielte den höchsten SARI-Wert unter den rein mittels Fine-Tuning optimierten

Modellen, was auf ein verbessertes Erlernen der relevanten Vereinfachungsoperationen hindeutet. Interessanterweise zeigten sich bei den Metriken BLEU und BERTScore nach dem Fine-Tuning leichte, modellabhängige Veränderungen. Während Gemma 3 4B mit Fine-Tuning seine bereits hohen Werte in diesen Bereichen beibehielt, sanken die Werte bei Qwen3 8B mit Fine-Tuning und Llama 3.1 8B mit Fine-Tuning im Vergleich zur jeweiligen satzweisen Basiskonfiguration geringfügig ab, verblieben jedoch auf einem absolut hohen Niveau. Dieser leichte Rückgang könnte als Indiz dafür gewertet werden, dass eine stärkere Spezialisierung auf den spezifischen Stil der Leichten Sprache zu minimalen Abweichungen von der N-Gramm-Struktur oder der exakten semantischen Abbildung der menschlichen Referenztexte führen kann. Dies muss nicht zwingend eine Verschlechterung der Textqualität bedeuten, sondern kann eine konsequentere Umsetzung der Vereinfachungsoperationen reflektieren, die von den menschlichen Referenztexten in dieser Form möglicherweise nicht immer eingehalten werden.

In-context Learning

Die Nutzung von ICL über ein RAG-System, bei dem dem Modell zur Inferenzzeit relevante Beispiele zur Verfügung gestellt werden, zeigte modellabhängige Stärken und verdeutlichte die Sensitivität dieser Methode. So konnte Qwen3 8B mit ICL seine bereits guten Leistungen im BLEU-Wert weiter ausbauen und den höchsten Durchschnittswert aller untersuchten SLMs Konfigurationen erreichen. Dies unterstreicht das Potenzial von ICL, die Generierung grammatikalisch und lexikalisch passender Sätze zu fördern, sofern das Modell für die präsentierten Beispiele empfänglich ist und diese die gewünschten stilistischen Merkmale adäquat repräsentieren. Auch der BERTScore von Qwen3 8B mit ICL verblieb auf einem hohen Niveau, was auf eine weiterhin hohe semantische Konsistenz hindeutet. Das Modell Llama 3.1 8B mit ICL erzielte den höchsten SARI-Wert innerhalb dieser Konfigurationsgruppe und übertraf damit sowohl Qwen3 8B mit ICL als auch Gemma 3 4B mit ICL. Dies legt nahe, dass Llama 3.1 8B besonders gut auf die durch die ICL-Beispiele demonstrierten Vereinfachungsoperationen reagierte. Dennoch zeigten die Ergebnisse keine großen Veränderungen der Werte gegenüber der Basiskonfiguration, außer bei der Einfachheit, wo jedoch unterschiedliche Ergebnisse auftraten. Während Llama 3.1 8B mit ICL und Gemma 3 4B mit ICL gute Flesch-Reading-Ease-Werte aufwiesen, die über ihren Basiswerten lagen, kam es bei Qwen3 8B mit ICL zu einem deutlichen Abfall des Werts im Vergleich zu seiner Basis- und insbesondere seiner Fine-Tuning Performance. Dieses Ergebnis ist bemerkenswert, lässt sich jedoch bei

genauerer Betrachtung der generierten Daten erklären. Anhand des folgenden Beispiels eines Textauszugs aus dem Datensatz wird deutlich, dass das Modell Qwen3 8B mit ICL in einigen Texten keine Satzenden generierten:

> „Tauchen Sie ein in ein Meer voller Farben
> Lassen Sie sich verzaubern
> Im Sea Life Hannover geht es auf eine Reise [...]“

Dies hatte zur Folge, dass der Flesch-Reading-Ease-Wert nicht ohne Weiteres präzise bestimmt werden konnte und Werte im Minusbereich erzeugt wurden. Dies erklärt den deutlich niedrigeren Durchschnittswert von 59.677. Der Medianwert von 69.990 ermöglicht in diesem Sinne eine realistischere Betrachtungsweise der Leistung, wobei dieser dennoch unter den Werten der anderen SLMs liegt.
Dies illustriert jedoch eine potenzielle Herausforderung des ICL-Ansatzes: Die Effektivität scheint einerseits von der Qualität der abgerufenen Beispiele abzuhängen. Andererseits kann die spezifischen Reaktion des Modells auf die Beispiele im System-Prompt und der Befolgung der Instruktionen ebenfalls von Bedeutung sein.

Kombination von Fine-Tuning und ICL

Die Kombination aus Fine-Tuning und ICL wurde untersucht, um potenzielle synergistische Effekte zu identifizieren. Hierbei zeigte sich, dass diese Kombination insbesondere die SARI-Werte weiter verbessern konnte. Gemma 3 4B mit Fine-Tuning und ICL erreichte den höchsten SARI-Wert aller untersuchten SLMs Konfigurationen, und auch Qwen3 8B mit Fine-Tuning und ICL erzielte einen hohen Wert. Dies könnte darauf hindeuten, dass ein bereits auf Leichte Sprache spezialisiertes Modell durch die zusätzlichen kontextuellen Beispiele des ICL noch präzisere und adäquatere Vereinfachungsoperationen erlernt oder anwendet. Hinsichtlich des Flesch-Reading-Ease-Wertes erzielte Qwen3 8B mit Fine-Tuning und ICL den höchsten Wert in dieser kombinierten Gruppe, verblieb jedoch unter den Werten aus dem reinen Fine-Tuning. Beim BLEU und BERTScore zeigte Gemma 3 4B mit Fine-Tuning und ICL die besten Werte, die mit seiner reinen Fine-Tuning Konfiguration vergleichbar waren.

Zusammenfassend lässt sich die zweite Forschungsfrage dahingehend beantworten, dass sowohl Fine-Tuning als auch ICL wirksame, jedoch unterschiedlich akzentuierte Optimierungsstrategien für SLMs im Kontext der Leichten Sprache darstellen kann. Fine-Tuning

erweist sich als besonders robust und effektiv zur konsistenten Steigerung der sprachlichen Einfachheit über verschiedene Modelle hinweg und kann zudem die Qualität der Vereinfachungsoperationen verbessern. ICL wiederum besitzt das Potenzial, spezifische Stärken einzelner Modelle weiter auszubauen, z.B. den BLEU-Wert bei Qwen3 8B mit ICL oder dem SARI-Wert bei Llama 3.1 8B mit ICL, zeigt jedoch auch eine höhere Sensitivität gegenüber der Interaktion zwischen Modell und dem System-Prompt, was sich in variableren Ergebnissen niederschlagen kann. Eine Kombination beider Ansätze kann, wie die SARI-Werte von Gemma 3 4B mit Fine-Tuning und ICL andeuten, zu den robustesten Ergebnissen bei spezifischen Vereinfachungsoperationen führen, ohne die durch Fine-Tuning erreichten Verbesserungen in anderen zentralen Bereichen signifikant zu schmälern. Die Wahl der optimalen Strategie oder deren Kombination hängt somit von den spezifischen Zielen und den Eigenschaften des eingesetzten Modells ab.

4.1.3 Vergleich von SLMs und SOTA-Modellen

Die dritte Forschungsfrage widmete sich dem Vergleich zwischen SLMs und den SOTA-Modellen. Dieser Vergleich erfolgte unter Berücksichtigung der Übersetzungsqualität sowie wirtschaftlicher und rechtlicher Aspekte, die insbesondere für den öffentlichen Sektor von Relevanz sein können.

Qualitative Anforderungen

Hinsichtlich der reinen Übersetzungsqualität ist, wie bereits geschildert, festzuhalten, dass proprietäre SOTA-Modelle, wie beispielsweise Gemini 2.5 Flash, in ihren Basiskonfigurationen oft eine hohe Ausgangsleistung bei der Generierung sprachlich einfacher Texte und bei der Durchführung spezifischer Vereinfachungsoperationen bieten. Diese höhere Grundfähigkeit könnte auf die umfangreichen und diversen Trainingsdaten sowie die potenziell fortgeschritteneren Architekturen dieser Modelle zurückzuführen sein. Die Ergebnisse dieser Arbeit demonstrieren jedoch, dass optimierte SLMs qualitativ nicht nur mithalten, sondern in spezifischen Bewertungsbereichen sogar überlegen sein können. Insbesondere durch Fine-Tuning konnten SLMs wie Qwen3 8B mit Fine-Tuning und Gemma 3 4B mit Fine-Tuning Flesch-Reading-Ease-Werte erzielen, die mit den führenden SOTA-Modellen konkurrieren oder diese im Median sogar übertreffen. Dies unterstreicht das Potenzial, auch kleinere, lokal betreibbare Modelle durch gezielte Anpassung auf ein sehr hohes Qualitätsniveau im Bereich der Textvereinfachung zu heben.

Interessanterweise zeigten die SLMs, allen voran Qwen3 8B, sowohl in der Basis- als auch in der ICL-optimierten Variante, bei Metriken wie BLEU und BERTScore oft eine höhere Fähigkeit oder zumindest Ebenbürtigkeit der Texterstellung gegenüber den untersuchten SOTA-Modellen. Dies könnte darauf hindeuten, dass sich SLMs tendenziell enger an der lexikalischen und semantischen Struktur der Referenzübersetzungen orientieren. Der Qualitätsunterschied ist somit nicht als absolut zu betrachten, sondern hängt stark von der betrachteten Metrik, dem spezifischen Modell und dem Grad der durchgeführten Optimierung ab. Einen Vergleich zwischen den generierten Texten von Qwen3 8B und Qwen3 8B mit Fine-Tuning und RAG sowie dem Gemini 2.5 Flash Modell und dem Referenzinput und -output, lässt sich unter A.13 einsehen.

Wirtschaftliche Anforderungen

Neben den qualitativen Aspekten spielen wirtschaftliche Überlegungen ebenfalls eine wesentliche Rolle, insbesondere im behördlichen Kontext. Um diesen Aspekt näher beleuchten zu können, wurden auf Basis der Evaluation die einzelnen Kosten pro Modell annäherungsweise ermittelt, welche sich unter A.14 einsehen lassen. Die Kosten wurden je nach Input- und Outputtoken aufgeschlüsselt. Dabei wurde zwischen dokumentbasiertem, satzbasiertem und satzbasiertem sowie mit ICL-Beispielen angereichertem Input unterschieden und in den einzelnen Tabellen A.11, A.12 sowie A.13 dargestellt. Die Anzahl der Inputtoken ergibt sich aus der Summe der Token der System-Prompts, siehe Implementierung A.9, und der Token der entsprechenden Dokumente sowie der Sätze mit und ohne Beispiele. Für die Berechnung der Outputtoken wurden alle Modellausgabetoken ebenfalls summiert. Der Einfachheit wurde für satzbasierten Input und den satzbasierten Input mit Beispielen dieselbe Anzahl an Outputtoken angenommen. Da aus den 100 Evaluationstexten insgesamt 1328 Sätze gebildet und jeweils in den System-Prompt eingebettet wurden, lässt sich die Inputtokenanzahl aus A.12 erklären. Zudem wurden die durchschnittliche Beispieltokenlänge aus dem TM berechnet, wodurch 217,2 Token je Anfrage in den System-Prompt mit ICL dazukamen, welche die Prompttoken aus A.13 ergeben. Zur Berechnung der Kosten wurden die API-Kosten auf OpenRouter[1] zu Grunde gelegt, die in der Tabelle A.10 für die ausgewählten Modelle dargestellt werden.

[1] https://openrouter.ai/models?fmt=table - Übersicht aller Modelle und ihrer Kosten - Abgerufen am 16.06.2025

Die Daten verdeutlichen erhebliche relative Unterschiede bei der Nutzung verschiedener LLMs für die Texterstellung. Die SLMs wie z.B. Gemma 3 4B und Llama 3.1 8B weisen durchweg sehr niedrige Inferenzkosten auf, die sich beispielsweise bei Llama 3.1 8B je nach Szenario zwischen 0,0023$ und 0,0135$ bewegen. Demgegenüber stehen deutlich höhere Kosten bei SOTA-Modellen wie DeepSeek V3-0324, bis zu 0,2424$, und insbesondere Gemini Flash 2.5 Pro, das bei dokumentbasiertem Input bereits insgesamt 4,26$ verursachte. Dies lag vor allem daran, dass durchschnittlich 3492 Reasoning Token pro Prompt ausgegeben worden sind, bevor die Übersetzung generiert wurde. Diese wurden in den Outputtoken in Tabelle A.11 berücksichtigt, wodurch sich die hohe Outputtokenanzahl ergibt. Diese Kostenrelationen bleiben auch dann bestehen, wenn unterschiedliche Input-Strategien (dokumentbasiert, satzbasiert, mit oder ohne ICL) zu variierenden absoluten Tokenzahlen und Gesamtkosten führen. Der Einsatz von SLMs ermöglicht somit nicht nur eine drastische Reduktion direkter Inferenz- und potenzieller Lizenzkosten im relativen Vergleich, insbesondere bei lokalem Betrieb ohne API-Nutzungsgebühren. Er eröffnet weiterhin Wege zu kosteneffizientem Fine-Tuning mittels ressourcenschonender Verfahren, was die Anpassung an spezifische Anwendungsfälle erleichtert und die Abhängigkeit von potenziell teureren externen Entwicklungen reduziert.
Insgesamt bewegen sich die Kosten für 100 Evaluationsbeispiele, mit Ausnahme des Gemini 2.5 Pro-Modells, im niedrigen Centbereich. Dies kann eine enorme Kostenreduktion für Übersetzungen im öffentlichen Sektor bedeuten. Gerade im Hinblick auf die genannten Zahlen von Asghari et al. von bis zu 4900 Euro pro Übersetzung einer Webseite [7], können LLMs unter diesen Umständen eine entscheidende Rolle spielen.

Rechtliche Anforderungen

Die in dieser Arbeit gezeigte hohe Leistungsfähigkeit optimierter SLMs gewinnt vor dem Hintergrund rechtlicher Überlegungen im öffentlichen Sektor zusätzlich an Bedeutung. Die Möglichkeit, Modelle lokal zu betreiben, und mittels ressourcenschonender Verfahren selbst anzupassen, adressiert zentrale Anforderungen an den Datenschutz und die Datenhoheit, die durch die DSGVO gefordert sind. Organisationen des öffentlichen Sektors behalten hierdurch die volle Kontrolle über die verarbeiteten, potenziell sensiblen Daten, was bei der Nutzung externer, proprietärer APIs, die oft Daten in andere Rechtsräume transferieren können, nicht im selben Maße gewährleistet ist. Hinsichtlich weiterer rechtlicher Anforderungen bei der Gestaltung der Leichten Sprache lassen sich jedoch wenige offizielle oder genormte Regelungen finden. Neben den inoffiziellen Regelwerken

stellt einzig Anlage 2 (zu § 3 Absatz 2) des BITV [27] eine Art Regelwerk dar, welches 13 Vorgaben enthält, auf deren Basis jene Texte überprüft werden könnten. Zudem könnte mit Hilfe der DIN SPEC 33429 [42] die Umsetzung der Texte in Leichte Sprache anhand konkreter Beispiele, ergänzend zur Anlage 2 (zu § 3 Absatz 2) des BITV, ebenfalls überprüft werden. Auf dieser Grundlage lassen sich die Vergleichstexte aus A.13 analysieren. Der Referenztext demonstriert die Einhaltung der Vorgaben nach Anlage 2 (zu § 3 Absatz 2) beispielsweise durch die Gliederung zusammengesetzter Wörter mittels Mittelpunkten und den Verzicht auf komplexe Grammatik wie Genitiv und Passiv. Unter den generierten Varianten lieferte Gemini 2.5 Flash einen qualitativ hochwertigen Text, der zusammengesetzte Wörter konsequent mit Bindestrichen versah und kurze, klar strukturierte Sätze bildete. Qwen3 8B mit Fine-Tuning und ICL setzte zwar ebenfalls den Mittelpunkt zur Wortgliederung ein, zeigte jedoch Defizite bei der Vermeidung des Genitivs und der konsistenten persönlichen Anrede. Die Basisversion Qwen3 8B wies die größten Abweichungen auf, etwa durch den Verzicht auf die Trennung von zusammengesetzten Wörtern und die Verwendung von Passivkonstruktionen. Dies demonstriert jedoch auch, dass durch die Nutzung der Optimierungsstrategien die einzelnen SLMs die Richtlinien besser umsetzen können.

Zusammenfassend lässt sich die dritte Forschungsfrage somit beantworten, dass lokal ausführbare SLMs, insbesondere nach gezielter Optimierung, eine leistungsstarke und strategisch vorteilhafte Alternative zu SOTA-LLMs für den Einsatz im öffentlichen Sektor darstellen. Die Qualitätsunterschiede in der Übersetzungsleistung sind, wie gezeigt, nicht unüberwindbar und können durch geeignete Maßnahmen oft ausgeglichen oder sogar zugunsten der SLMs verschoben werden. Die Vorteile hinsichtlich Datenschutz, Datenkontrolle, Anpassbarkeit und potenzieller Kosteneffizienz, können für den öffentlichen Sektor von erheblicher Relevanz sein. Die Entscheidung für ein bestimmtes Modell oder eine bestimmte Modellklasse sollte daher auf einer sorgfältigen Abwägung der spezifischen Gewichtung der Metriken und der genannten nicht-funktionalen Anforderungen basieren.

4.2 Einordnung der Ergebnisse in den Forschungskontext

Die in dieser Arbeit erzielten Ergebnisse können einen Beitrag zum aktuellen Forschungsstand der monolingualen Maschinenübersetzung von Standardsprache in Leichte Sprache

leisten. Sie bestätigen nicht nur bestehende Herausforderungen, sondern demonstrieren auch die Wirksamkeit spezifischer Anpassungstechniken für SLMs im Low-Resource Kontext.

Ein erstes Ergebnis ist die Bestätigung der von Schwarzer und Kauchak [135] beschriebenen negativen Korrelation zwischen der Einfachheit eines Textes und seiner semantischen Ähnlichkeit zum Original. In den untersuchten Konfigurationen führte eine Steigerung des Flesch-Reading-Ease-Wertes konsistent zu einer Verringerung des BERTScores und umgekehrt. Dies unterstreicht die anhaltende Herausforderung, die Aspekte Einfachheit und Semantik gleichzeitig zu optimieren.
Darüber hinaus liefern die spezifischen Modellleistungen detaillierte Einblicke. Die Stärken der Basismodelle Qwen3 8B mit hohen BLEU und BERTScore-Werten und Gemma 3 4B mit guter Flesch-Reading-Ease Leistung verdeutlichen, dass aktuelle SLMs unterschiedliche Eignungen für die Aufgabe der Textvereinfachung aufweisen. Dies legt nahe, dass die Wahl des Basismodells sowie spezifische Anpassungen entscheidend ist, um den Anforderungen der Leichten Sprache gerecht zu werden. Dieser Aspekt ergänzt die generelle Forschung zur Anpassung von Modellen an spezifische Zielsprachen und -stile [167, 174].
Die deutliche Verbesserung der Flesch-Reading-Ease-Werte durch gezieltes Fine-Tuning bei allen evaluierten SLMs, besonders bei den Varianten Qwen3 8B mit Fine-Tuning und Gemma 3 4B mit Fine-Tuning, untermauert die Effektivität des Style-Trainings. Dieser Befund steht im Einklang mit den von Anschütz et al. [5] für frühere Modellgenerationen beschriebenen Ansätzen und bestätigt zudem deren Übertragbarkeit auf neuere Modelle. Weiterhin erweist sich, angesichts der fehlenden Standardisierung der Leichten Sprache [43] und der Einstufung als Low-Resource Sprache [97], Fine-Tuning als eine valide Strategie, um Modelle an die spezifischen stilistischen Merkmale anzupassen. Dies gilt auch in Bezug auf Ansätzen wie LoRA für ein ressourcenschonendes Training, mit hoher Effektivität insbesondere im Low-Resource Kontext [70, 86].
Außerdem stützen die positiven Auswirkungen des RAG-basierten ICL, vor allem die Steigerung der BLEU-Werte bei Qwen3 8B mit ICL und der SARI-Werte bei Llama 3.1 8B mit ICL, die Thesen von Kopp et al. [83] zum vielversprechenden Einsatz hybrider Techniken beim Einsatz eines TM mit LLMs. Die Fähigkeit von RAG relevante Beispiele zur Laufzeit bereitzustellen, kann die Qualität der generierten Texte verbessern, was im Einklang mit Studien steht, die den Nutzen von RAG und ICL für Low-Resource Sprachen und neuronalen Methoden hervorheben [69, 106, 108, 163, 180].

Zusammenfassend zeigen die Ergebnisse, dass sowohl Fine-Tuning als auch RAG-basiertes ICL wertvolle Methoden zur Anpassung von SLMs für die Übersetzung in Leichte Sprache darstellen. Während Fine-Tuning besonders zur stilistischen Anpassung beiträgt, kann RAG mit ICL die kontextuelle Relevanz und entsprechende Wortwahl verbessern. Dies deckt sich mit der allgemeinen Beobachtung, dass beide Ansätze die Leistung von LLMs verbessern können, wobei die Wahl von der spezifischen Aufgabe und den verfügbaren Ressourcen abhängt [88, 117].

4.3 Limitationen

Neben den erzielten Ergebnissen dieser Arbeit mit den vorliegenden Methoden, wurden einige Faktoren bei der Untersuchung der generierten Texte der LLMs und Optimierungsstrategien nicht berücksichtigt.

Zum Einen sind in dieser Arbeit lediglich zwei Optimierungsstrategien betrachtet worden, Fine-Tuning und ICL. Nach Anisuzzaman [4] lassen sich noch weitere Strategien anwenden, darunter z.B. Prefix-, Prompt- oder Hyperparametertuning. Zu Unterscheiden ist Prompt-Tuning zudem vom Prompt-Engineering, bei der die Anweisungen an das Modell selbst angepasst werden. In dieser Arbeit wurden infolgedessen keine verschiedenen System-Prompts untersucht, sondern stets dieselbe Anweisung verwendet, die abhängig von der genutzten Optimierungsstrategie war. Einen solchen Prompt-Engineering Ansatz in Bezug auf Low-Resource Sprachen, auch in Verbindung mit Fine-Tuning, lässt sich beispielsweise bei Khoboko et al. [80] finden.
Die Wirksamkeit des Fine-Tunings und des RAG-Systems hängt zudem stark von der Qualität und dem Umfang des verwendeten Datensatzes ab. Obwohl ein Datensatz mit 5643 TUs für das TM und 500 TUs für das Fine-Tuning erstellt wurde, könnten spezifische Eigenheiten dieses Datensatzes die Leistung und Generalisierbarkeit der Modelle beeinflussen. In diesem Zusammenhang wurden z.B. keine verschiedenen Größen von TMs untersucht. Zudem wurde auch nicht berücksichtigt, wie hoch der Einfluss einer unterschiedlicher Anzahl an Trainingsbeispielen für das Fine-Tuning auf die Leistung von SLMs zur Generierung von Texten in Leichter Sprache hat. Außerdem stellt die Auswahl von drei SLMs und drei SOTA-Modellen lediglich eine Stichprobe dar. Andere Modelle derselben Familien oder von anderen Anbietern könnten abweichende Ergebnisse zeigen.

Weiterhin wurde in dieser Arbeit die Qualität des Retrievals des RAG-Systems nicht isoliert evaluiert, sondern lediglich die Übersetzungen der Modelle. Dies kann die Resultate ebenfalls in gewisser Weise beeinflusst haben, obwohl bei der Implementation des Systems darauf geachtet wurde, dass die Retrieval-Beispiele eine gewisse Relevanz zum zu übersetzenden Text haben. Abschließend ist die Bewertung der Ergebnisse nur anhand automatisierter Metriken erfolgt. Menschliche Expertenbeurteilungen der Texte könnten bessere Aufklärung darüber geben, ob die Qualität der generierten Texte tatsächlich den Anforderungen an die Leichte Sprache genügen.

5 Fazit

5.1 Zusammenfassung

Die vorliegende Arbeit untersuchte die Eignung von LLMs zur monolingualen Maschinenübersetzung von Standardsprache in Leichte Sprache, speziell zugeschnitten auf die Anforderungen des öffentlichen Sektors in Deutschland. Im Fokus standen dabei die grundlegenden Fähigkeiten von LLMs, die Wirksamkeit von Optimierungsstrategien sowie der Vergleich von SLMs mit oft proprietären SOTA-Modellen unter Berücksichtigung qualitativer, wirtschaftlicher und rechtlicher Aspekte. Die zentralen Erkenntnisse der Arbeit werden nachfolgend durch die Beantwortung der Forschungsfragen dargelegt:

1. **Inwieweit sind LLMs in der Lage, automatisiert Texte in Leichter Sprache für den öffentlichen Sektor zu erstellen, die den Kriterien zur Bewertung der Leichten Sprache hinsichtlich Einfachheit, Grammatikalität und Semantik entsprechen?**
   Die Untersuchung zeigte, dass LLMs grundsätzlich über diese Fähigkeit verfügen. Insbesondere SOTA-Modelle lieferten bereits in ihren Basiskonfigurationen Ergebnisse von hoher sprachlicher Einfachheit. Gleichzeitig konnten auch kleinere, quelloffene SLMs überzeugende Resultate hinsichtlich der Kriterien Einfachheit, Grammatikalität und semantischer Ähnlichkeit erzielen, was ihre grundlegende Eignung für die Aufgabe bestätigt.

2. **Können die Optimierungsstrategien ICL mittels RAG und Fine-Tuning die Leistung von LLMs bei der automatischen Erstellung von Texten in Leichter Sprache im öffentlichen Sektor verbessern?**
   Es konnte nachgewiesen werden, dass Optimierungsstrategien die Leistung von SLMs signifikant verbessern. Fine-Tuning erwies sich als besonders robuste Methode zur Steigerung der sprachlichen Einfachheit und zur gezielten Umsetzung von

Vereinfachungsoperationen. ICL mittels RAG zeigte ebenfalls Potenzial zur Stärkung modellspezifischer Stärken, wies jedoch eine höhere Sensitivität und variablere Ergebnisse auf. Die Kombination beider Ansätze deutete auf synergistische Effekte hin, die sich insbesondere in einer verbesserten qualitativen Bewertung zeigten.

3. **Inwieweit lassen sich LLMs mit kleinerer Parameterzahl im Vergleich zu LLMs mit hoher Parameterzahl bzw. SOTA-Modellen effektiv für die automatische Erstellung von Leichter Sprache im öffentlichen Sektor einsetzen, unter Berücksichtigung von qualitativen, wirtschaftlichen und rechtlichen Anforderungen an die Übersetzungen?**
   Im direkten Vergleich wurde deutlich, dass optimierte SLMs eine leistungsstarke und strategisch vorteilhafte Alternative für den öffentlichen Sektor darstellen können. Während SOTA-Modelle eine hohe Ausgangsleistung zeigten, konnten optimierte SLMs qualitativ mithalten und in spezifischen Metriken sogar überlegen sein. Der entscheidende Vorteil der SLMs liegt jedoch in den nicht-qualitativen Aspekten. Ihre Fähigkeit zum lokalen Betrieb gewährleistet Datenschutzkonformität und Datenhoheit. Zudem bieten sie eine höhere Anpassbarkeit und potenziell erheblich geringere Inferenzkosten. Diese Faktoren machen sie für den kosten- und datensensiblen öffentlichen Sektor besonders attraktiv und ermöglichen eine bessere Umsetzung von Richtlinien wie der BITV und der DSGVO.

Zusammenfassend lässt sich festhalten, dass die automatisierte Erstellung von Leichter Sprache mittels LLMs, insbesondere durch den gezielten Einsatz und die Optimierung von SLMs, einen vielversprechenden Weg darstellt, um den qualitativen, wirtschaftlichen und rechtlichen Anforderungen im öffentlichen Sektor gerecht zu werden. Die Ergebnisse unterstreichen, dass eine sorgfältige Auswahl des Modells und der Optimierungsstrategien, unter Berücksichtigung der spezifischen Rahmenbedingungen, essenziell für einen erfolgreichen Einsatz ist.

5.2 Ausblick

Aufbauend auf den Ergebnissen und Limitationen dieser Arbeit ergeben sich mehrere Richtungen für zukünftige Forschungsvorhaben. Zukünftige Arbeiten könnten die Ursachen für die unterschiedliche Reaktion der Modelle auf ICL genauer untersuchen, beispielsweise durch eine Analyse der Qualität und Relevanz der abgerufenen Beispiele für

jedes Modell und jede Metrik. Dies schließt auch die isolierte Evaluation der Retrieval-Qualität des RAG-Systems ein. Eine Erweiterung der Arbeit um eine systematische menschliche Evaluation mithilfe von Likert-Skalen der von den Modellkonfigurationen generierten Texte wäre entscheidend, um die Ergebnisse der automatischen Metriken zu validieren, die tatsächliche Verständlichkeit zu prüfen und die Nutzerakzeptanz zu bewerten. Dies könnte mithilfe der DIN SPEC 33429 anhand konkreter Beispiele in Zusammenarbeit mit der Zielgruppe erfolgen. Interessant wäre ebenfalls eine Untersuchung von iterativen Optimierungsstrategien, bei der beispielsweise ein Modell genutzt wird, um die Beispiele für das RAG-System zu verbessern oder neue synthetische Daten für weiteres Fine-Tuning zu generieren, die zu weiteren Qualitätssteigerungen führen könnten. Darüber hinaus wäre die Untersuchung weiterer Optimierungsstrategien wie Prefix-Tuning, Prompt-Tuning oder spezialisiertes Prompt-Engineering von Interesse, um das Potenzial von SLMs weiter auszuschöpfen. Die Rolle der Datensatzgröße und -qualität sowohl für das Fine-Tuning als auch für die Effektivität des RAG-Systems bedarf weiterer Untersuchungen, um optimale Konfigurationen für Low-Resource Szenarien zu ermitteln. Die Evaluation könnte auf eine breitere Auswahl von SLMs und SOTA-Modellen ausgeweitet werden, um die Generalisierbarkeit der Ergebnisse zu stärken und möglicherweise noch besser geeignete Modelle zu identifizieren. Weiterhin wäre die Entwicklung und Integration von Mechanismen, die eine strengere Einhaltung spezifischer Regeln der Leichten Sprache, wie sie beispielsweise in Anlage 2 (zu § 3 Absatz 2) der BITV oder in der DIN SPEC 33429 formuliert sind, während des Generierungsprozesses ebenfalls eine Möglichkeit, die Qualität der Texte zu verbessern. Dies könnte beispielsweise durch das genannte Prompt-Engineering oder auch durch Post-Processing der generierten Texte durch andere Modelle erfolgen. Schließlich könnte die Entwicklung spezifischerer, automatisierter Metriken, die die Nuancen der Leichten Sprache besser erfassen, die Evaluation weiter verfeinern und objektiver gestalten, indem beispielsweise vorhandene Metriken miteinander kombiniert werden. Auch eine detailliertere Analyse der Wirtschaftlichkeit, die über die reinen Inferenzkosten hinausgeht und Aspekte wie Wartung, Hardware und Schulungsaufwand für lokale Modelle berücksichtigt, könnte für den öffentlichen Sektor wertvoll sein.

Literatur

[1] N. A. Alfear et al., „Meta-Evaluation of Sentence Simplification Metrics“, in „Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)“, N. Calzolari et al., Hrsg., Torino, Italia: ELRA und ICCL, Mai 2024, S. 11 229–11 235. Adresse: https://aclanthology.org/2024.lrec-main.981/ (Abgerufen am 12. 03. 2025).

[2] S. Alissa und M. Wald, „Text simplification using transformer and BERT“, en, „Computers, Materials & Continua“, Jg. 75, Nr. 2, S. 3479–3495, 2023, „Num Pages: 17 Number: 2“. DOI: 10.32604/cmc.2023.033647. Adresse: https://eprints.soton.ac.uk/477287/ (Abgerufen am 06. 01. 2025).

[3] F. Alva-Manchego et al., „The (Un)Suitability of Automatic Evaluation Metrics for Text Simplification“, „Computational Linguistics“, Jg. 47, Nr. 4, S. 861–889, Dez. 2021, ISSN: 0891-2017. DOI: 10.1162/coli_a_00418. Adresse: https://doi.org/10.1162/coli_a_00418 (Abgerufen am 12. 03. 2025).

[4] D. M. Anisuzzaman et al., „Fine-Tuning Large Language Models for Specialized Use Cases“, „Mayo Clinic Proceedings: Digital Health“, Jg. 3, Nr. 1, S. 100 184, März 2025, ISSN: 2949-7612. DOI: 10.1016/j.mcpdig.2024.11.005. Adresse: https://www.sciencedirect.com/science/article/pii/S2949761224001147 (Abgerufen am 06. 01. 2025).

[5] M. Anschütz et al., „Language Models for German Text Simplification: Overcoming Parallel Data Scarcity through Style-specific Pre-training“, in „Findings of the Association for Computational Linguistics: ACL 2023“, „arXiv:2305.12908 [cs]“, 2023, S. 1147–1158. DOI: 10.18653/v1/2023.findings-acl.74. Adresse: http://arxiv.org/abs/2305.12908 (Abgerufen am 06. 01. 2025).

[6] S. Arle und C. Frondén, „Bringing order to chaos: Research on Easy Swedish“, en, „Nordic Journal of Linguistics“, Jg. 45, Nr. 2, S. 167–193, Okt. 2022, ISSN: 0332-5865, 1502-4717. DOI: 10.1017/S0332586522000105. Adresse: https://www.cambridge.org/core/journals/nordic-journal-of-linguistics/article/bringing-order-to-chaos-research-on-easy-swedish/B92A3E6521727FDBF999DFC468A445A6 (Abgerufen am 05. 03. 2025).

[7] H. Asghari et al., „On the Prevalence of Leichte Sprache on the German Web“, in „Proceedings of the 15th ACM Web Science Conference 2023“, Ser. WebSci '23, New York, NY, USA: Association for Computing Machinery, Apr. 2023, S. 147–152, ISBN: 979-8-4007-0089-7. DOI: 10.1145/3578503.3583599. Adresse: https://dl.acm.org/doi/10.1145/3578503.3583599 (Abgerufen am 20. 03. 2025).

[8] J. L. Ba et al., „Layer Normalization“, „arXiv:1607.06450 [stat]“, Juli 2016. DOI: 10.48550/arXiv.1607.06450. Adresse: http://arxiv.org/abs/1607.06450 (Abgerufen am 15. 03. 2025).

[9] D. Bahdanau et al., „Neural Machine Translation by Jointly Learning to Align and Translate“, „arXiv:1409.0473 [cs]“, Mai 2016. DOI: 10.48550/arXiv.1409.0473. Adresse: http://arxiv.org/abs/1409.0473 (Abgerufen am 10. 03. 2025).

[10] A. Balaguer et al., „RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture“, „arXiv:2401.08406 [cs] version: 2“, Jan. 2024. DOI: 10.48550/arXiv.2401.08406. Adresse: http://arxiv.org/abs/2401.08406 (Abgerufen am 07. 02. 2025).

[11] A. Baumert, „Leichte Sprache - Einfache Sprache: Literaturrecherche, Interpretation, Entwicklung“, de. Hochschule Hannover, 2016, „Artwork Size: 12727 KB, 293 pages Medium: application/pdf Pages: 12727 KB, 293 pages“. DOI: 10.25968/OPUS-697. Adresse: https://serwiss.bib.hs-hannover.de/697 (Abgerufen am 05. 03. 2025).

[12] Y. Belkada et al., „Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA“, „Abgerufen am 10.01.2025“, 2023. Adresse: https://huggingface.co/blog/4bit-transformers-bitsandbytes (Abgerufen am 10. 01. 2025).

[13] R. Bhagdev et al., „Hybrid Search: Effectively Combining Keywords and Semantic Searches“, en, in „The Semantic Web: Research and Applications“, S. Bechhofer et al., Hrsg., Berlin, Heidelberg: Springer, 2008, S. 554–568, ISBN: 978-3-540-68234-9. DOI: 10.1007/978-3-540-68234-9_41.

[14] B. M. Bock et al., „Leichte Sprache, einfache Sprache, verständliche Sprache“ („Narr Studienbücher“), ger. Tübingen: Narr Francke Attempto, 2021, ISBN: 978-3-8233-8181-5.

[15] B. Bogale et al., „RAG Based QA for Low Resource Languages“, Nov. 2024. DOI: 10.21203/rs.3.rs-5360450/v1. Adresse: https://www.researchsquare.com/article/rs-5360450/v1 (Abgerufen am 21. 03. 2025).

[16] R.-M. Botarleanu et al., „Sequence-to-Sequence Models for Automated Text Simplification“, en, in „Artificial Intelligence in Education“, I. I. Bittencourt et al., Hrsg., Cham: Springer International Publishing, 2020, S. 31–36, ISBN: 978-3-030-52240-7. DOI: 10.1007/978-3-030-52240-7_6.

[17] B. Boyce, „Beyond topicality: A two stage view of relevance and the retrieval process“, „Information Processing & Management“, Jg. 18, Nr. 3, S. 105–109, Jan. 1982, ISSN: 0306-4573. DOI: 10.1016/0306-4573(82)90033-4. Adresse: https://www.sciencedirect.com/science/article/pii/0306457382900334 (Abgerufen am 27. 05. 2025).

[18] L. Boytsov et al., „Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search“, in „Proceedings of the 25th ACM International on Conference on Information and Knowledge Management“, Ser. CIKM '16, New York, NY, USA: Association for Computing Machinery, Okt. 2016, S. 1099–1108, ISBN: 978-1-4503-4073-1. DOI: 10.1145/2983323.2983815. Adresse: https://doi.org/10.1145/2983323.2983815 (Abgerufen am 18. 03. 2025).

[19] U. Bredel und C. Maaß, „Leichte Sprache: Theoretische Grundlagen Orientierung für die Praxis“, ger. Berlin: Duden Verlag, 2016, ISBN: 978-3-411-91178-3.

[20] P. F. Brown et al., „Aligning sentences in parallel corpora“, in „29th annual meeting of the association for computational linguistics“, 1991, S. 169–176.

[21] P. F. Brown et al., „A statistical approach to machine translation“, „Comput. Linguist.“, Jg. 16, Nr. 2, S. 79–85, Juni 1990, ISSN: 0891-2017.

[22] P. F. Brown et al., „The mathematics of statistical machine translation: parameter estimation“, „Comput. Linguist.“, Jg. 19, Nr. 2, S. 263–311, Juni 1993, ISSN: 0891-2017.

[23] T. Brown et al., „Language Models are Few-Shot Learners“, in „Advances in Neural Information Processing Systems“, Bd. 33, Curran Associates, Inc., 2020, S. 1877–1901. Adresse: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (Abgerufen am 16. 03. 2025).

[24] Bundesamt der Justiz, „BDSG - Bundesdatenschutzgesetz“, 2017. Adresse: https://www.gesetze-im-internet.de/bdsg_2018/BJNR209710017.html (Abgerufen am 03. 03. 2025).

[25] Bundesamt der Justiz, „BGG - Gesetz zur Gleichstellung von Menschen mit Behinderungen“, 2022. Adresse: https://www.gesetze-im-internet.de/bgg/BJNR146800002.html (Abgerufen am 28. 02. 2025).

[26] Bundesamt der Justiz, „BITV 2.0 - Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz“, 2023. Adresse: https://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html (Abgerufen am 28. 02. 2025).

[27] Bundesamt der Justiz, „Anlage 2 BITV 2.0 - Einzelnorm“. Adresse: https://www.gesetze-im-internet.de/bitv_2_0/anlage_2.html (Abgerufen am 06. 03. 2025).

[28] Bundesministerium für wirtschaftliche Zusammenarbeit und Entwicklung, „Übereinkommen über die Rechte von Menschen mit Behinderungen (UN-Behindertenrechtskonvention)“, de. Adresse: https://www.bmz.de/de/service/lexikon/uebereinkommen-ueber-die-rechte-von-menschen-mit-behinderungen-60274 (Abgerufen am 05. 03. 2025).

[29] Bundesregierung, „Interview zum Tag der Leichten Sprache | Bundesregierung“, de, Mai 2023. Adresse: https://www.bundesregierung.de/breg-de/aktuelles/interview-anne-leichtfuss-1918176 (Abgerufen am 04. 01. 2025).

[30] Bundesregierung, „15 Jahre UN-Behindertenrechtskonvention“, de, März 2024. Adresse: https://www.bundesregierung.de/breg-de/aktuelles/un-behindertenrechtskonvention-2261648 (Abgerufen am 05. 03. 2025).

[31] D. Cai et al., „Neural Machine Translation with Monolingual Translation Memory“, in „Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)“, C. Zong et al., Hrsg., Online: Association for Computational Linguistics, Aug. 2021, S. 7307–7318. DOI: 10.18653/v1/2021.acl-long.567. Adresse: https://aclanthology.org/2021.acl-long.567/ (Abgerufen am 03. 02. 2025).

[32] A. Cenić et al., „The Serbian Retrieval Augmented Generation System based on Hybrid Search“, in „2024 International Conference Automatics and Informatics (ICAI)“, Okt. 2024, S. 420–424. DOI: 10.1109/ICAI63388.2024.10851665. Adresse: https://ieeexplore.ieee.org/document/10851665 (Abgerufen am 27. 05. 2025).

[33] D. Chen et al., „Reading Wikipedia to Answer Open-Domain Questions“, in „Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)“, R. Barzilay und M.-Y. Kan, Hrsg., Vancouver, Canada: Association for Computational Linguistics, Juli 2017, S. 1870–1879. DOI: 10.18653/v1/P17-1171. Adresse: https://aclanthology.org/P17-1171/ (Abgerufen am 18. 03. 2025).

[34] K. Cho et al., „Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation“, in „Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)“, A. Moschitti et al., Hrsg., Doha, Qatar: Association for Computational Linguistics, Okt. 2014, S. 1724–1734. DOI: 10.3115/v1/D14-1179. Adresse: https://aclanthology.org/D14-1179/ (Abgerufen am 10. 03. 2025).

[35] T. P. Christensen und A. Schjoldager, „Translation-Memory (TM) Research: What Do We Know and How Do We Know It?“ en, „HERMES - Journal of Language and Communication in Business“, Nr. 44, S. 89–101, Okt. 2010, „Number: 44“, ISSN: 1903-1785. DOI: 10.7146/hjlcb.v23i44.97268. Adresse: https://tidsskrift.dk/her/article/view/97268 (Abgerufen am 09. 03. 2025).

[36] W. Coster und D. Kauchak, „Learning to Simplify Sentences Using Wikipedia“, in „Proceedings of the Workshop on Monolingual Text-To-Text Generation“, K. Filippova und S. Wan, Hrsg., Portland, Oregon: Association for Computational Linguistics, Juni 2011, S. 1–9. Adresse: https://aclanthology.org/W11-1601/ (Abgerufen am 12. 03. 2025).

[37] DeepSeek-AI et al., „DeepSeek-V3 Technical Report“, „arXiv:2412.19437 [cs]“, Feb. 2025. DOI: 10.48550/arXiv.2412.19437. Adresse: http://arxiv.org/abs/2412.19437 (Abgerufen am 23. 05. 2025).

[38] Der Beauftragte der Bundesregierung für die Belange von Menschen mit Behinderungen, „Behindertengleichstellungsgesetz“, de. Adresse: http://www.behindertenbeauftragter.de/DE/AS/rechtliches/behindertengleichstellungsgesetz/behindertengleichstellungsgesetz.html?nn=26550 (Abgerufen am 05. 03. 2025).

[39] T. Dettmers et al., „QLORA: efficient finetuning of quantized LLMs“, in „Proceedings of the 37th International Conference on Neural Information Processing Systems“, Ser. NIPS '23, Red Hook, NY, USA: Curran Associates Inc., Mai 2024, S. 10 088–10 115. (Abgerufen am 09. 01. 2025).

[40] J. Devlin et al., „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“, in „Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)“, J. Burstein et al., Hrsg., Minneapolis, Minnesota: Association for Computational Linguistics, Juni 2019, S. 4171–4186. DOI: 10.18653/v1/N19-1423. Adresse: https://aclanthology.org/N19-1423/ (Abgerufen am 16. 03. 2025).

[41] DFL, „„Laut Studien profitieren bundesweit mehr als 16 Millionen Menschen von Leichter Sprache““, de, Mai 2023. Adresse: https://www.dfl.de/de/aktuelles/laut-studien-profitieren-bundesweit-mehr-als-16-millionen-menschen-von-leichter-sprache/ (Abgerufen am 04. 01. 2025).

[42] DIN-SPEC Konsortium, „DIN SPEC 33429:2025-03, Empfehlungen für Deutsche Leichte Sprache“, 2025. DOI: 10.31030/3594547. Adresse: https://www.dinmedia.de/de/-/-/387728031 (Abgerufen am 16. 06. 2025).

[43] DIN-SPEC-Konsortium, „DIN SPEC 33429:2023-04, Empfehlungen für Deutsche Leichte Sprache“, 2023. DOI: 10.31030/3417293. Adresse: https://www.dinmedia.de/de/-/-/364785446 (Abgerufen am 06. 01. 2025).

[44] N. Ding et al., „Parameter-efficient fine-tuning of large-scale pre-trained language models“, en, „Nature Machine Intelligence“, Jg. 5, Nr. 3, S. 220–235, März 2023, „Publisher: Nature Publishing Group“, ISSN: 2522-5839. DOI: 10.1038/s42256-

023-00626-4. Adresse: https://www.nature.com/articles/s42256-023-00626-4 (Abgerufen am 17. 03. 2025).

[45] Q. Dong et al., „A Survey on In-context Learning“, in „Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing“, Y. Al-Onaizan et al., Hrsg., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, S. 1107–1128. DOI: 10.18653/v1/2024.emnlp-main.64. Adresse: https://aclanthology.org/2024.emnlp-main.64/ (Abgerufen am 03. 02. 2025).

[46] S. Ebling et al., „Automatic Text Simplification for German“, English, „Frontiers in Communication“, Jg. 7, Feb. 2022, „Publisher: Frontiers“, ISSN: 2297-900X. DOI: 10.3389/fcomm.2022.706718. Adresse: https://www.frontiersin.org/journals/communication/articles/10.3389/fcomm.2022.706718/full (Abgerufen am 06. 01. 2025).

[47] I. Espinosa-Zaragoza et al., „A Review of Research-Based Automatic Text Simplification Tools“, in „Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing“, R. Mitkov und G. Angelova, Hrsg., Varna, Bulgaria: INCOMA Ltd., Shoumen, Bulgaria, Sep. 2023, S. 321–330. Adresse: https://aclanthology.org/2023.ranlp-1.36/ (Abgerufen am 03. 03. 2025).

[48] Europäischen Union, „Richtlinie (EU) 2016/2102 des Europäischen Parlaments und des Rates vom 26. Oktober 2016 über den barrierefreien Zugang zu den Websites und mobilen Anwendungen öffentlicher Stellen“, de, 2016. Adresse: https://eur-lex.europa.eu/eli/dir/2016/2102/oj/deu (Abgerufen am 28. 02. 2025).

[49] Europäischen Union, „Verordnung - 2016/679 - DE - Datenschutz Grundverordnung - EUR-Lex“, de, „Doc ID: 32016R0679 Doc Sector: 3 Doc Title: Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung) (Text von Bedeutung für den EWR) Doc Type: R Usr_lan: de“, 2016. Adresse: https://eur-lex.europa.eu/eli/reg/2016/679/oj/deu (Abgerufen am 03. 03. 2025).

[50] M. A. Fărcașiu et al., „Easy-to-Read: Evolution and Perspectives—A Bibliometric Analysis of Research, 1978–2021“, en, „International Journal of Environmen-

tal Research and Public Health“, Jg. 20, Nr. 4, S. 3359, Jan. 2023, „Number: 4 Publisher: Multidisciplinary Digital Publishing Institute“, ISSN: 1660-4601. DOI: 10.3390/ijerph20043359. Adresse: https://www.mdpi.com/1660-4601/20/4/3359 (Abgerufen am 05. 03. 2025).

[51] R. Flesch, „A new readability yardstick“, „Journal of Applied Psychology“, Jg. 32, Nr. 3, S. 221–233, 1948, „Place: US Publisher: American Psychological Association“, ISSN: 1939-1854. DOI: 10.1037/h0057532.

[52] G. Freyhoff et al., „Sag es einfach! Europäische Richtlinien für die Erstellung von leicht lesbaren Informationen für Menschen mit geistiger Behinderung“, 1998. Adresse: https://www.web-4-all.de/wp-content/uploads/2012/12/EURichtlinie_sag_es_einfach.pdf (Abgerufen am 05. 03. 2025).

[53] K. P. Gaffney et al., „SQLite: past, present, and future“, „Proc. VLDB Endow.“, Jg. 15, Nr. 12, S. 3535–3547, Aug. 2022, ISSN: 2150-8097. DOI: 10.14778/3554821.3554842. Adresse: https://doi.org/10.14778/3554821.3554842 (Abgerufen am 05. 02. 2025).

[54] P. Gage, „A new algorithm for data compression“, „The C Users Journal archive“, Feb. 1994. Adresse: https://www.semanticscholar.org/paper/A-new-algorithm-for-data-compression-Gage/1aa9c0045f1fe8c79cce03c7c14ef4b4643a21f8 (Abgerufen am 14. 03. 2025).

[55] W. A. Gale, K. W. Church et al., „A program for aligning sentences in bilingual corpora“, „Computational linguistics“, Jg. 19, Nr. 1, S. 75–102, 1994.

[56] Y. Gao et al., „Retrieval-Augmented Generation for Large Language Models: A Survey“, 2023, „Publisher: arXiv Version Number: 5“. DOI: 10.48550/ARXIV.2312.10997. Adresse: https://arxiv.org/abs/2312.10997 (Abgerufen am 03. 02. 2025).

[57] M. Glass et al., „Re2G: Retrieve, Rerank, Generate“, in „Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies“, M. Carpuat et al., Hrsg., Seattle, United States: Association for Computational Linguistics, Juli 2022, S. 2701–2715. DOI: 10.18653/v1/2022.naacl-main.194. Adresse: https://aclanthology.org/2022.naacl-main.194/ (Abgerufen am 27. 05. 2025).

[58] N. Grabar und H. Saggion, „Evaluation of Automatic Text Simplification: Where are we now, where should we go from here“, in „Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale“, Y. Estève et al., Hrsg., Avignon, France: ATALA, Juni 2022, S. 453–463. Adresse: https://aclanthology.org/2022.jeptalnrecital-taln.47/ (Abgerufen am 06. 01. 2025).

[59] A. Grattafiori et al., „The Llama 3 Herd of Models“, „Version Number: 3“, 2024. DOI: 10.48550/ARXIV.2407.21783. Adresse: https://arxiv.org/abs/2407.21783 (Abgerufen am 11. 07. 2025).

[60] A. Grotlüschen et al., „LEO 2018 - Living with Low Literacy“, de, 2021. DOI: 10.4232/1.13771. Adresse: https://www.fachportal-paedagogik.de/literatur/vollanzeige.html?FId=3380475 (Abgerufen am 04. 01. 2025).

[61] Hamburgische Beauftragten für Datenschutz und Informationsfreiheit, „Diskussionspapier: Large Language Models und personenbezogene Daten“, 2024. Adresse: https://datenschutz-hamburg.de/fileadmin/user_upload/HmbBfDI/Datenschutz/Informationen/240715_Diskussionspapier_HmbBfDI_KI_Modelle.pdf (Abgerufen am 03. 03. 2025).

[62] D. Han und M. Han, „unslothai/unsloth“, „Abgerufen am 08.01.2025“, 2023. Adresse: https://github.com/unslothai/unsloth (Abgerufen am 08. 01. 2025).

[63] B. C. Harris, „Bi-text, a new concept in translation theory“, 1988. Adresse: https://api.semanticscholar.org/CorpusID:117398176.

[64] C. Harsha et al., „Synthetic Data Generation Using Large Language Models for Financial Question Answering“, in „Proceedings of the Joint Workshop of the 9th Financial Technology and Natural Language Processing (FinNLP), the 6th Financial Narrative Processing (FNP), and the 1st Workshop on Large Language Models for Finance and Legal (LLMFinLegal)“, C.-C. Chen et al., Hrsg., Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, S. 76–95. Adresse: https://aclanthology.org/2025.finnlp-1.7/ (Abgerufen am 27. 05. 2025).

[65] K. He et al., „Deep Residual Learning for Image Recognition“, in „2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)“, „ISSN: 1063-6919“, Juni 2016, S. 770–778. DOI: 10.1109/CVPR.2016.90. Adresse: https://ieeexplore.ieee.org/document/7780459 (Abgerufen am 15. 03. 2025).

[66] Q. He et al., „Fast and Accurate Neural Machine Translation with Translation Memory“, in „Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)“, C. Zong et al., Hrsg., Online: Association for Computational Linguistics, Aug. 2021, S. 3170–3180. DOI: 10.18653/v1/2021.acl-long.246. Adresse: https://aclanthology.org/2021.acl-long.246/ (Abgerufen am 03. 02. 2025).

[67] S. Hochreiter und J. Schmidhuber, „Long Short-Term Memory“, en, „Neural Computation“, Jg. 9, Nr. 8, S. 1735–1780, Nov. 1997, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. Adresse: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (Abgerufen am 13. 03. 2025).

[68] J. Hoffmann et al., „Training compute-optimal large language models“, in „Proceedings of the 36th International Conference on Neural Information Processing Systems“, Ser. NIPS '22, Red Hook, NY, USA: Curran Associates Inc., Nov. 2022, S. 30 016–30 030, ISBN: 978-1-7138-7108-8. (Abgerufen am 16. 03. 2025).

[69] S. B. Hosseinbeigi et al., „Advancing Retrieval-Augmented Generation for Persian: Development of Language Models, Comprehensive Benchmarks, and Best Practices for Optimization“, „arXiv:2501.04858 [cs]“, Jan. 2025. DOI: 10.48550/arXiv.2501.04858. Adresse: http://arxiv.org/abs/2501.04858 (Abgerufen am 21. 03. 2025).

[70] E. J. Hu et al., „LoRA: Low-Rank Adaptation of Large Language Models“, „arXiv:2106.09685 [cs]“, Okt. 2021. DOI: 10.48550/arXiv.2106.09685. Adresse: http://arxiv.org/abs/2106.09685 (Abgerufen am 07. 01. 2025).

[71] Huggingface.co, „bitsandbytes“, „Abgerufen am 10.01.2025“. Adresse: https://huggingface.co/docs/bitsandbytes/main/en/index (Abgerufen am 10. 01. 2025).

[72] E. Husel, „Leichte Sprache in der Bundesverwaltung“, de, in „Leichte Sprache in der Bundesverwaltung: Was? Wer? Wie?“ E. Husel, Hrsg., Berlin: Frank & Timme GmbH, 2022, S. 53–72, ISBN: 978-3-7329-9057-3. DOI: 10.57088/978-3-7329-

9057-3_4. Adresse: https://doi.org/10.57088/978-3-7329-9057-3_4 (Abgerufen am 25. 02. 2025).

[73] J. Hutchins, „The first public demonstration of machine translation : the Georgetown-IBM system , 7 th January 1954“, 2006. Adresse: https://open.unive.it/hitrade/books/HutchinsFirst.pdf (Abgerufen am 07. 03. 2025).

[74] Inclusion Europe, „Informationen für alle: europäische Regeln, wie man Informationen leicht lesbar und leicht verständlich macht ; [entwickelt im Rahmen des Projektes Pathways - Wege zur Erwachsenenbildung für Menschen mit Lernschwierigkeiten]“, de. Brüssel: Inclusion Europe, 2009, „OCLC: 838006206“, ISBN: 978-2-87460-111-8.

[75] Jina AI, „Ich bin ein Berliner: Deutsch-Englische Bilinguale Embeddings mit 8K Token-Länge“, de, „Abgerufen am 04.02.2025“, Jan. 2024. Adresse: https://jina.ai/news/ich-bin-ein-berliner-german-english-bilingual-embeddings-with-8k-token-length (Abgerufen am 04. 02. 2025).

[76] Jina AI, „Jina Embeddings v3: Ein wegweisendes mehrsprachiges Embedding-Modell“, de, „Abgerufen am 04.02.2025“, Sep. 2024. Adresse: https://jina.ai/news/jina-embeddings-v3-a-frontier-multilingual-embedding-model (Abgerufen am 04. 02. 2025).

[77] D. Jurafsky und J. H. Martin, „Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models“, 3rd. 2025. Adresse: https://web.stanford.edu/~jurafsky/slp3/ (Abgerufen am 10. 03. 2025).

[78] Z. Ke et al., „Continual Pre-training of Language Models“, „arXiv:2302.03241 [cs]“, Apr. 2023. DOI: 10.48550/arXiv.2302.03241. Adresse: http://arxiv.org/abs/2302.03241 (Abgerufen am 17. 03. 2025).

[79] G. Kellermann, „Leichte und Einfache Sprache – Versuch einer Definition“, de, Feb. 2014. Adresse: https://www.bpb.de/shop/zeitschriften/apuz/179341/leichte-und-einfache-sprache-versuch-einer-definition/ (Abgerufen am 03. 03. 2025).

[80] P. W. Khoboko et al., „Optimizing translation for low-resource languages: Efficient fine-tuning with custom prompt engineering in large language models“, „Machine Learning with Applications“, Jg. 20, S. 100 649, Juni 2025, ISSN: 2666-

8270. DOI: 10.1016/j.mlwa.2025.100649. Adresse: https://www.sciencedirect.com/science/article/pii/S2666827025000325 (Abgerufen am 04. 06. 2025).

[81] L. Klöser et al., „German Text Simplification: Finetuning Large Language Models with Semi-Synthetic Data“, „arXiv:2402.10675 [cs]“, Feb. 2024. DOI: 10.48550/arXiv.2402.10675. Adresse: http://arxiv.org/abs/2402.10675 (Abgerufen am 06. 01. 2025).

[82] P. Koehn, „Statistical machine translation“, eng. Cambridge New York: Cambridge University Press, 2010, ISBN: 978-0-511-81582-9 978-0-511-68910-9 978-0-511-69132-4.

[83] T. Kopp et al., „Towards machine translation into Easy Language in public administrations“, en, in „Emerging Fields in Easy Language and Accessible Communication Research“, S. Deilen et al., Hrsg., Berlin: Frank & Timme GmbH, 2023, S. 371–406, ISBN: 978-3-7329-9026-9. DOI: 10.57088/978-3-7329-9026-9_14. Adresse: https://doi.org/10.57088/978-3-7329-9026-9_14 (Abgerufen am 20. 03. 2025).

[84] P. Lewis et al., „Retrieval-augmented generation for knowledge-intensive NLP tasks“, in „Proceedings of the 34th International Conference on Neural Information Processing Systems“, Ser. NIPS '20, Red Hook, NY, USA: Curran Associates Inc., Dez. 2020, S. 9459–9474, ISBN: 978-1-7138-2954-6. (Abgerufen am 03. 02. 2025).

[85] X. Liang et al., „Multi-Lingual Sentence Alignment with GPT Models“, in „2023 4th International Conference on Artificial Intelligence and Data Sciences (AiDAS)“, Sep. 2023, S. 218–223. DOI: 10.1109/AiDAS60501.2023.10284652. Adresse: https://ieeexplore.ieee.org/document/10284652 (Abgerufen am 29. 03. 2025).

[86] X. Liang et al., „Toward Low-Resource Languages Machine Translation: A Language-Specific Fine-Tuning With LoRA for Specialized Large Language Models“, „IEEE Access“, Jg. 13, S. 46 616–46 626, 2025, „Conference Name: IEEE Access“, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3549795. Adresse: https://ieeexplore.ieee.org/document/10918960 (Abgerufen am 21. 03. 2025).

[87] T. Lin et al., „A survey of transformers“, „AI Open“, Jg. 3, S. 111–132, Jan. 2022, ISSN: 2666-6510. DOI: 10.1016/j.aiopen.2022.10.001. Adres-

se: https://www.sciencedirect.com/science/article/pii/S2666651022000146 (Abgerufen am 15. 03. 2025).

[88] H. Liu et al., „Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning“, 2022, „Publisher: arXiv Version Number: 2“. DOI: 10.48550/ARXIV.2205.05638. Adresse: https://arxiv.org/abs/2205.05638 (Abgerufen am 03. 02. 2025).

[89] L. Long et al., „On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey“, in „Findings of the Association for Computational Linguistics: ACL 2024“, L.-W. Ku et al., Hrsg., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, S. 11 065–11 082. DOI: 10.18653/v1/2024.findings-acl.658. Adresse: https://aclanthology.org/2024.findings-acl.658/ (Abgerufen am 27. 05. 2025).

[90] Y. Lu et al., „Understanding and Improving Transformer From a Multi-Particle Dynamic System Point of View“, „arXiv:1906.02762 [cs]“, Juni 2019. DOI: 10.48550/arXiv.1906.02762. Adresse: http://arxiv.org/abs/1906.02762 (Abgerufen am 15. 03. 2025).

[91] Z. Lu et al., „Small Language Models: Survey, Measurements, and Insights“, „arXiv:2409.15790 [cs]“, Feb. 2025. DOI: 10.48550/arXiv.2409.15790. Adresse: http://arxiv.org/abs/2409.15790 (Abgerufen am 23. 05. 2025).

[92] C. Maaß, „Leichte Sprache: das Regelbuch“ („Barrierefreie Kommunikation“ Bd. 1), ger. Berlin Münster: LIT, 2015, ISBN: 978-3-643-12907-9.

[93] C. Maaß, „Easy Language – Plain Language – Easy Language Plus: Balancing Comprehensibility and Acceptability“ („Easy – Plain – Accessible“), eng. Berlin: Frank & Timme, 2020, ISBN: 978-3-7329-0691-8 978-3-7329-9299-7.

[94] C. Maaß und I. Rink, Hrsg., „Handbuch Barrierefreie Kommunikation“. Frank & Timme, 2018, ISBN: 978-3-7329-0407-5. DOI: 10.26530/20.500.12657/43216. Adresse: https://library.oapen.org/handle/20.500.12657/43216 (Abgerufen am 03. 03. 2025).

[95] M. Maddela et al., „LENS: A Learnable Evaluation Metric for Text Simplification“, in „Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)“, A. Rogers et al., Hrsg., Toronto, Canada: Association for Computational Linguistics, Juli 2023, S. 16 383–16 408. DOI: 10.18653/v1/2023.acl-long.905. Adresse: https://aclanthology.org/2023.acl-long.905/ (Abgerufen am 21. 03. 2025).

[96] M. Madina et al., „Easy-to-Read in Germany: A Survey on its Current State and Available Resources“, „arXiv:2306.03189 [cs]“, Juni 2023. DOI: 10.48550/arXiv.2306.03189. Adresse: http://arxiv.org/abs/2306.03189 (Abgerufen am 20. 03. 2025).

[97] A. Magueresse et al., „Low-resource Languages: A Review of Past Work and Future Challenges“, „ArXiv“, Juni 2020. Adresse: https://www.semanticscholar.org/paper/Low-resource-Languages%3A-A-Review-of-Past-Work-and-Magueresse-Carles/4de3595439ed8e433e19979b49ea9171c01dc846 (Abgerufen am 03. 02. 2025).

[98] L. Martin et al., „Reference-less Quality Estimation of Text Simplification Systems“, en, Nov. 2018. Adresse: https://inria.hal.science/hal-01959054 (Abgerufen am 06. 03. 2025).

[99] T. Maruyama und K. Yamamoto, „Extremely Low Resource Text simplification with Pre-trained Transformer Language Model“, in „2019 International Conference on Asian Language Processing (IALP)“, Nov. 2019, S. 53–58. DOI: 10.1109/IALP48816.2019.9037650. Adresse: https://ieeexplore.ieee.org/abstract/document/9037650 (Abgerufen am 06. 01. 2025).

[100] R. Merx et al., „Low-Resource Machine Translation through Retrieval-Augmented LLM Prompting: A Study on the Mambai Language“, „arXiv:2404.04809 [cs]“, Apr. 2024. DOI: 10.48550/arXiv.2404.04809. Adresse: http://arxiv.org/abs/2404.04809 (Abgerufen am 21. 03. 2025).

[101] T. Mikolov et al., „Efficient Estimation of Word Representations in Vector Space“, „arXiv:1301.3781 [cs]“, Sep. 2013. DOI: 10.48550/arXiv.1301.3781. Adresse: http://arxiv.org/abs/1301.3781 (Abgerufen am 14. 03. 2025).

[102] S. Minaee et al., „Large Language Models: A Survey“, „arXiv:2402.06196 [cs]“, Feb. 2024. DOI: 10.48550/arXiv.2402.06196. Adresse: http://arxiv.org/abs/2402.06196 (Abgerufen am 06. 01. 2025).

[103] T. Miyazaki et al., „Understanding How Positional Encodings Work in Transformer Model“, in „Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)“, N. Calzolari et al., Hrsg., Torino, Italia: ELRA und ICCL, Mai 2024, S. 17 011–17 018. Adresse: https://aclanthology.org/2024.lrec-main.1478/ (Abgerufen am 15. 03. 2025).

[104] I. Mohr et al., „Multi-Task Contrastive Learning for 8192-Token Bilingual Text Embeddings“, „arXiv:2402.17016 [cs]“, Feb. 2024. DOI: 10.48550/arXiv.2402.17016. Adresse: http://arxiv.org/abs/2402.17016 (Abgerufen am 04. 02. 2025).

[105] A. Monem et al., „Generating Arabic text in multilingual speech-to-speech machine translation framework“, „Machine Translation“, Jg. 22, S. 205–258, Dez. 2008. DOI: 10.1007/s10590-009-9054-9.

[106] Y. Mu et al., „Augmenting Large Language Model Translators via Translation Memories“, in „Findings of the Association for Computational Linguistics: ACL 2023“, A. Rogers et al., Hrsg., Toronto, Canada: Association for Computational Linguistics, Juli 2023, S. 10 287–10 299. DOI: 10.18653/v1/2023.findings-acl.653. Adresse: https://aclanthology.org/2023.findings-acl.653/ (Abgerufen am 09. 06. 2025).

[107] M. Nagao, „A framework of a mechanical translation between Japanese and English by analogy principle“, in „Proc. of the international NATO symposium on Artificial and human intelligence“, USA: Elsevier North-Holland, Inc., Okt. 1984, S. 173–180, ISBN: 978-0-444-86545-8. (Abgerufen am 09. 03. 2025).

[108] Z. A. Nazi et al., „Evaluation of open and closed-source LLMs for low-resource language with zero-shot, few-shot, and chain-of-thought prompting“, „Natural Language Processing Journal“, Jg. 10, S. 100 124, März 2025, ISSN: 2949-7191. DOI: 10.1016/j.nlp.2024.100124. Adresse: https://www.sciencedirect.com/science/article/pii/S2949719124000724 (Abgerufen am 03. 02. 2025).

[109] E. V. Ndimbo et al., „Leveraging Retrieval-Augmented Generation for Swahili Language Conversation Systems“, en, „Applied Sciences“, Jg. 15, Nr. 2, S. 524, Jan. 2025, „Number: 2 Publisher: Multidisciplinary Digital Publishing Institute“, ISSN: 2076-3417. DOI: 10.3390/app15020524. Adresse: https://www.mdpi.com/2076-3417/15/2/524 (Abgerufen am 21. 03. 2025).

[110] Netzwerk Leichte Sprache e.V, „Was ist das Netzwerk?“ Adresse: https://www.netzwerk-leichte-sprache.de/ls/das-netzwerk/was-ist-das-netzwerk (Abgerufen am 05. 03. 2025).

[111] Netzwerk Leichte Sprache e.V., „Die Regeln für Leichte Sprache vom Netzwerk Leichte Sprache“, „Abgerufen am 06.01.2025“, 2022. Adresse: https://www.

netzwerk-leichte-sprache.de/fileadmin/content/documents/regeln/Regelwerk_NLS_Neuauflage-2022.pdf.

[112] Netzwerk People First Deutschland e.V., „Verein – Mensch zuerst“, de. Adresse: https://www.menschzuerst.de/wer-sind-wir/verein/ (Abgerufen am 05. 03. 2025).

[113] G. Neubig, „Neural Machine Translation and Sequence-to-sequence Models: A Tutorial“, „ArXiv“, Jg. abs/1703.01619, 2017. (Abgerufen am 05. 01. 2025).

[114] C. V. Nguyen et al., „A Survey of Small Language Models“, „arXiv:2410.20011 [cs]“, Okt. 2024. DOI: 10.48550/arXiv.2410.20011. Adresse: http://arxiv.org/abs/2410.20011 (Abgerufen am 23. 05. 2025).

[115] N. I. Nikolov und R. Hahnloser, „Large-Scale Hierarchical Alignment for Data-driven Text Rewriting“, in „Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)“, R. Mitkov und G. Angelova, Hrsg., Varna, Bulgaria: INCOMA Ltd., Sep. 2019, S. 844–853. DOI: 10.26615/978-954-452-056-4_098. Adresse: https://aclanthology.org/R19-1098/ (Abgerufen am 21. 03. 2025).

[116] S. Nisioi et al., „Exploring Neural Text Simplification Models“, in „Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)“, R. Barzilay und M.-Y. Kan, Hrsg., Vancouver, Canada: Association for Computational Linguistics, Juli 2017, S. 85–91. DOI: 10.18653/v1/P17-2014. Adresse: https://aclanthology.org/P17-2014/ (Abgerufen am 12. 03. 2025).

[117] O. Ovadia et al., „Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs“, in „Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing“, Y. Al-Onaizan et al., Hrsg., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, S. 237–250. DOI: 10.18653/v1/2024.emnlp-main.15. Adresse: https://aclanthology.org/2024.emnlp-main.15/ (Abgerufen am 21. 03. 2025).

[118] K. Papineni et al., „Bleu: a Method for Automatic Evaluation of Machine Translation“, in „Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics“, P. Isabelle et al., Hrsg., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Juli 2002, S. 311–318. DOI: 10.3115/1073083.1073135. Adresse: https://aclanthology.org/P02-1040 (Abgerufen am 16. 12. 2024).

[119] R. Pascanu et al., „On the difficulty of training recurrent neural networks“, en, in „Proceedings of the 30th International Conference on Machine Learning“, „ISSN: 1938-7228“, PMLR, Mai 2013, S. 1310–1318. Adresse: https://proceedings.mlr.press/v28/pascanu13.html (Abgerufen am 13. 03. 2025).

[120] P. Patwa et al., „Enhancing Low-Resource LLMs Classification with PEFT and Synthetic Data“, „arXiv:2404.02422 [cs]“, Apr. 2024. DOI: 10.48550/arXiv.2404.02422. Adresse: http://arxiv.org/abs/2404.02422 (Abgerufen am 10. 02. 2025).

[121] T. Poibeau, „Machine translation“ („The MIT Press essential knowledge series“), eng. Cambridge, Massachusetts London, England: The MIT Press, 2017, ISBN: 978-0-262-34243-8.

[122] A. Radford et al., „Language Models are Unsupervised Multitask Learners“, 2019. Adresse: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe (Abgerufen am 03. 02. 2025).

[123] C. Raffel et al., „Exploring the limits of transfer learning with a unified text-to-text transformer“, „J. Mach. Learn. Res.“, Jg. 21, Nr. 1, 140:5485–140:5551, Jan. 2020, ISSN: 1532-4435.

[124] I. Rink und C. Maaß, „Verständlichkeit und Gesundheitskompetenz im Spektrum zwischen Leichter und Einfacher Sprache“, de, in „Gesundheitskompetenz“, K. Rathmann et al., Hrsg., Berlin, Heidelberg: Springer, 2023, S. 487–499, ISBN: 978-3-662-67055-2. DOI: 10.1007/978-3-662-67055-2_146. Adresse: https://doi.org/10.1007/978-3-662-67055-2_146 (Abgerufen am 04. 03. 2025).

[125] S. Robertson, „Understanding inverse document frequency: on theoretical arguments for IDF“, „Journal of Documentation“, Jg. 60, Nr. 5, S. 503–520, Jan. 2004, „Publisher: Emerald Group Publishing Limited“, ISSN: 0022-0418. DOI: 10.1108/00220410410560582. Adresse: https://doi.org/10.1108/00220410410560582 (Abgerufen am 18. 03. 2025).

[126] S. E. Robertson et al., „Okapi at TREC-3“, en, 1994. Adresse: http://trec.nist.gov/pubs/trec3/papers/city.ps.gz (Abgerufen am 18. 03. 2025).

[127] R. K. S, „Small Language Models and Their Role in Hybrid AI Architectures for Big Data Analytics“, in „2024 International Conference on Sustainable Communication Networks and Application (ICSCNA)“, Dez. 2024, S. 594–599. DOI: 10.1109/ICSCNA63714.2024.10863995. Adresse: https://ieeexplore.ieee.org/document/10863995 (Abgerufen am 03. 03. 2025).

[128] V. Sadler und R. Vendelmans, „Pilot Implementation of a Bilingual Knowledge Bank“, in „COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics“, 1990. Adresse: https://aclanthology.org/C90-3101/ (Abgerufen am 09. 03. 2025).

[129] S. Samsi et al., „From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference“, in „2023 IEEE High Performance Extreme Computing Conference (HPEC)“, „ISSN: 2643-1971“, Sep. 2023, S. 1–9. DOI: 10.1109/HPEC58863.2023.10363447. Adresse: https://ieeexplore.ieee.org/abstract/document/10363447 (Abgerufen am 03. 03. 2025).

[130] A. Säuberli et al., „Benchmarking Data-driven Automatic Text Simplification for German“, eng, in „Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)“, N. Gala und R. Wilkens, Hrsg., Marseille, France: European Language Resources Association, Mai 2020, S. 41–48, ISBN: 979-10-95546-45-0. Adresse: https://aclanthology.org/2020.readi-1.7/ (Abgerufen am 06. 01. 2025).

[131] S. Schmutz et al., „Easy-to-read language in disability-friendly web sites: Effects on nondisabled users“, „Applied Ergonomics“, Jg. 74, S. 97–106, Jan. 2019, ISSN: 0003-6870. DOI: 10.1016/j.apergo.2018.08.013. Adresse: https://www.sciencedirect.com/science/article/pii/S0003687018302862 (Abgerufen am 02. 03. 2025).

[132] M. Schnebbe, „Large Language Models und personenbezogene Daten“, de, „Datenschutz und Datensicherheit - DuD“, Jg. 48, Nr. 12, S. 797–799, Dez. 2024, ISSN: 1862-2607. DOI: 10.1007/s11623-024-2020-0. Adresse: https://doi.org/10.1007/s11623-024-2020-0 (Abgerufen am 03. 03. 2025).

[133] T. Schomacker et al., „Data and Approaches for German Text simplification – towards an Accessibility-enhanced Communication“, „arXiv:2312.09966 [cs]“, Dez. 2023. DOI: 10.48550/arXiv.2312.09966. Adresse: http://arxiv.org/abs/2312.09966 (Abgerufen am 06. 01. 2025).

[134] N. Schümann, „Exkurs: Large Language Models“, de, in „Gamechanger Künstliche Intelligenz: Wie künstliche Intelligenz inspiriert und kreatives Potenzial entfesselt“, N. Schümann, Hrsg., München: Haufe, 2024, S. 31–39, ISBN: 978-3-648-17563-7. DOI: 10.34157/978-3-648-17563-7_4. Adresse: https://doi.org/10.34157/978-3-648-17563-7_4 (Abgerufen am 03. 03. 2025).

[135] M. Schwarzer, „Human Evaluation for Text Simplification : The Simplicity-Adequacy Tradeoff“, 2018. Adresse: https://www.semanticscholar.org/paper/Human-Evaluation-for-Text-Simplification-%3A-The-Schwarzer/76d7f28362f81be856ef38c142ec9b78d154088b (Abgerufen am 12. 03. 2025).

[136] R. Sennrich und M. Volk, „MT-based Sentence Alignment for OCR-generated Parallel Texts“, in „Proceedings of the 9th Conference of the Association for Machine Translation in the Americas: Research Papers“, Denver, Colorado, USA: Association for Machine Translation in the Americas, Okt. 2010. Adresse: https://aclanthology.org/2010.amta-papers.14/ (Abgerufen am 10. 03. 2025).

[137] R. Sennrich et al., „Neural Machine Translation of Rare Words with Subword Units“, in „Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)“, K. Erk und N. A. Smith, Hrsg., Berlin, Germany: Association for Computational Linguistics, Aug. 2016, S. 1715–1725. DOI: 10.18653/v1/P16-1162. Adresse: https://aclanthology.org/P16-1162/ (Abgerufen am 14. 03. 2025).

[138] M. Shao et al., „Survey of different Large Language Model Architectures: Trends, Benchmarks, and Challenges“, „IEEE Access“, Jg. 12, S. 188 664–188 706, 2024, „arXiv:2412.03220 [cs]“, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3482107. Adresse: http://arxiv.org/abs/2412.03220 (Abgerufen am 15. 03. 2025).

[139] H. Somers und M. Díaz, „Translation Memory vs. Example-based MT – What's the difference?“, 2004. Adresse: https://www.semanticscholar.org/paper/Translation-Memory-vs.-Example-based-MT-%E2%80%93-What%E2%80%99s-Somers-D%C3%ADaz/f057423da01a9499c1331d67689c96c89f406842 (Abgerufen am 09. 03. 2025).

[140] H. Somers, „Review Article: Example-based Machine Translation“, en, „Machine Translation“, Jg. 14, Nr. 2, S. 113–157, Juni 1999, ISSN: 1573-0573. DOI: 10.

1023/A:1008109312730. Adresse: https://doi.org/10.1023/A:1008109312730 (Abgerufen am 09. 03. 2025).

[141] H. Soudani et al., „Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge“, in „Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region“, „arXiv:2403.01432 [cs]“, Dez. 2024, S. 12–22. DOI: 10.1145/3673791.3698415. Adresse: http://arxiv.org/abs/2403.01432 (Abgerufen am 05. 02. 2025).

[142] L. Specia, „Translating from Complex to Simplified Sentences“, en, in „Computational Processing of the Portuguese Language“, T. A. S. Pardo et al., Hrsg., Berlin, Heidelberg: Springer, 2010, S. 30–39, ISBN: 978-3-642-12320-7. DOI: 10.1007/978-3-642-12320-7_5.

[143] N. Spring et al., „Analyzing sentence alignment for automatic simplification of German texts“, en, in „Emerging Fields in Easy Language and Accessible Communication Research“, S. Deilen et al., Hrsg., Berlin: Frank & Timme GmbH, 2023, S. 339–369, ISBN: 978-3-7329-9026-9. DOI: 10.57088/978-3-7329-9026-9_13. Adresse: https://doi.org/10.57088/978-3-7329-9026-9_13 (Abgerufen am 20. 03. 2025).

[144] S. Stajner, „Automatic Text Simplification for Social Good: Progress and Challenges“, in „Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021“, C. Zong et al., Hrsg., Online: Association for Computational Linguistics, Aug. 2021, S. 2637–2652. DOI: 10.18653/v1/2021.findings-acl.233. Adresse: https://aclanthology.org/2021.findings-acl.233/ (Abgerufen am 03. 03. 2025).

[145] R. Stodden et al., „DEplain: A German Parallel Corpus with Intralingual Translations into Plain Language for Sentence and Document Simplification“, in „Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)“, A. Rogers et al., Hrsg., Toronto, Canada: Association for Computational Linguistics, Juli 2023, S. 16 441–16 463. DOI: 10.18653/v1/2023.acl-long.908. Adresse: https://aclanthology.org/2023.acl-long.908/ (Abgerufen am 06. 01. 2025).

[146] S. Sturua et al., „jina-embeddings-v3: Multilingual Embeddings With Task LoRA“, „arXiv:2409.10173 [cs]“, Sep. 2024. DOI: 10.48550/arXiv.2409.10173. Adresse: http://arxiv.org/abs/2409.10173 (Abgerufen am 04. 02. 2025).

[147] T. Su et al., „Unlocking Parameter-Efficient Fine-Tuning for Low-Resource Language Translation“, in „Findings of the Association for Computational Linguistics: NAACL 2024“, K. Duh et al., Hrsg., Mexico City, Mexico: Association for Computational Linguistics, Juni 2024, S. 4217–4225. DOI: 10.18653/v1/2024.findings-naacl.263. Adresse: https://aclanthology.org/2024.findings-naacl.263/ (Abgerufen am 21. 03. 2025).

[148] J. Suter et al., „Rule-based Automatic Text Simplification for German“, s.n., Sep. 2016. DOI: 10.5167/UZH-128601. Adresse: https://www.zora.uzh.ch/id/eprint/128601 (Abgerufen am 06. 01. 2025).

[149] I. Sutskever et al., „Sequence to sequence learning with neural networks“, in „Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2“, Ser. NIPS'14, Bd. 2, Cambridge, MA, USA: MIT Press, Dez. 2014, S. 3104–3112. (Abgerufen am 10. 03. 2025).

[150] G. Team et al., „Gemma 3 Technical Report“, „arXiv:2503.19786 [cs]“, März 2025. DOI: 10.48550/arXiv.2503.19786. Adresse: http://arxiv.org/abs/2503.19786 (Abgerufen am 23. 05. 2025).

[151] S. S. Al-Thanyyan und A. M. Azmi, „Automated Text Simplification: A Survey“, „ACM Comput. Surv.“, Jg. 54, Nr. 2, 43:1–43:36, März 2021, ISSN: 0360-0300. DOI: 10.1145/3442695. Adresse: https://dl.acm.org/doi/10.1145/3442695 (Abgerufen am 06. 01. 2025).

[152] R. Theja, „Boosting RAG: Picking the Best Embedding & Reranker models — LlamaIndex - Build Knowledge Assistants over your Enterprise Data“, en, „Abgerufen am 04.02.2025“, 2023. Adresse: https://www.llamaindex.ai/blog/boosting-rag-picking-the-best-embedding-reranker-models-42d079022e83 (Abgerufen am 04. 02. 2025).

[153] B. Thompson und P. Koehn, „Vecalign: Improved Sentence Alignment in Linear Time and Space“, in „Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)“, K. Inui et al., Hrsg., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, S. 1342–1348. DOI: 10.18653/v1/D19-1136. Adresse: https://aclanthology.org/D19-1136/ (Abgerufen am 10. 03. 2025).

[154] P. Tillet et al., „Triton: an intermediate language and compiler for tiled neural network computations“, in „Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages“, Ser. MAPL 2019, New York, NY, USA: Association for Computing Machinery, Juni 2019, S. 10–19, ISBN: 978-1-4503-6719-6. DOI: 10.1145/3315508.3329973. Adresse: https://doi.org/10.1145/3315508.3329973 (Abgerufen am 09. 01. 2025).

[155] V. Toborek et al., „A New Aligned Simple German Corpus“, in „Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)“, A. Rogers et al., Hrsg., Toronto, Canada: Association for Computational Linguistics, Juli 2023, S. 11 393–11 412. DOI: 10.18653/v1/2023.acl-long.638. Adresse: https://aclanthology.org/2023.acl-long.638/ (Abgerufen am 07. 01. 2025).

[156] P. Toma, „Systran as a multilingual machine translation system“, in „Proceedings of the third european congress on information systems and networks, overcoming the language barrier“, 1977, S. 569–581.

[157] A. Vaswani et al., „Decoding with Large-Scale Neural Language Models Improves Translation“, S. 1387–1392, 2013. (Abgerufen am 05. 01. 2025).

[158] A. Vaswani et al., „Attention Is All You Need“, „arXiv:1706.03762 [cs]“, Aug. 2023. DOI: 10.48550/arXiv.1706.03762. Adresse: http://arxiv.org/abs/1706.03762 (Abgerufen am 10. 03. 2025).

[159] B. Vauquois, „A survey of formal grammars and algorithms for recognition and transformation in mechanical translation“, in „IFIP congress (2)“, „tex.citedby: 0 tex.cites: 0“, 1968, S. 1114–1122.

[160] C. Wang et al., „Cost-Effective Hyperparameter Optimization for Large Language Model Generation Inference“, en, in „Proceedings of the Second International Conference on Automated Machine Learning“, „ISSN: 2640-3498“, PMLR, Dez. 2023, S. 21/1–17. Adresse: https://proceedings.mlr.press/v224/wang23b.html (Abgerufen am 03. 03. 2025).

[161] F. Wang et al., „A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness“, „arXiv:2411.03350 [cs]“, Dez. 2024. DOI: 10.48550/arXiv.2411.03350. Adresse: http://arxiv.org/abs/2411.03350 (Abgerufen am 23. 05. 2025).

[162] H. Wang et al., „Progress in Machine Translation“, „Engineering“, Jg. 18, S. 143–153, Nov. 2022, ISSN: 2095-8099. DOI: 10.1016/j.eng.2021.03.023. Adresse: https://www.sciencedirect.com/science/article/pii/S2095809921002745 (Abgerufen am 10. 03. 2025).

[163] J. Wang et al., „Retrieval-Augmented Machine Translation with Unstructured Knowledge“, „arXiv:2412.04342 [cs]“, Dez. 2024. DOI: 10.48550/arXiv.2412.04342. Adresse: http://arxiv.org/abs/2412.04342 (Abgerufen am 03. 02. 2025).

[164] T. Wang et al., „An Experimental Study of LSTM Encoder-Decoder Model for Text Simplification“, „arXiv:1609.03663 [cs]“, Sep. 2016. DOI: 10.48550/arXiv.1609.03663. Adresse: http://arxiv.org/abs/1609.03663 (Abgerufen am 12. 03. 2025).

[165] X. Wang et al., „Searching for Best Practices in Retrieval-Augmented Generation“, in „Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing“, Y. Al-Onaizan et al., Hrsg., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, S. 17 716–17 736. DOI: 10.18653/v1/2024.emnlp-main.981. Adresse: https://aclanthology.org/2024.emnlp-main.981/ (Abgerufen am 18. 03. 2025).

[166] W. Weaver, „Translation“, in „Machine translation of languages“, W. N. Locke und A. D. Boothe, Hrsg., Cambridge, MA: MIT Press, 1949, S. 15–23. Adresse: https://www.cs.cmu.edu/~leili/course/dl4mt21fa/Weaver_1949_Translation.pdf (Abgerufen am 07. 03. 2025).

[167] M. Wu et al., „Adapting Large Language Models for Document-Level Machine Translation“, „ArXiv“, Jg. abs/2401.06468, 2024. DOI: 10.48550/arXiv.2401.06468. (Abgerufen am 05. 01. 2025).

[168] X. Wu und Y. Arase, „An In-depth Evaluation of Large Language Models in Sentence Simplification with Error-based Human Assessment“, „Version Number: 3“, 2024. DOI: 10.48550/ARXIV.2403.04963. Adresse: https://arxiv.org/abs/2403.04963 (Abgerufen am 07. 06. 2025).

[169] W. Xu et al., „Problems in Current Text Simplification Research: New Data Can Help“, „Transactions of the Association for Computational Linguistics“, Jg. 3, M. Collins und L. Lee, Hrsg., S. 283–297, 2015, „Place: Cambridge, MA Publisher: MIT Press“. DOI: 10.1162/tacl_a_00139. Adresse: https://aclanthology.org/Q15-1021/ (Abgerufen am 12. 03. 2025).

[170] W. Xu et al., „Optimizing Statistical Machine Translation for Text Simplification“, „Transactions of the Association for Computational Linguistics“, Jg. 4, L. Lee et al., Hrsg., S. 401–415, 2016, „Place: Cambridge, MA Publisher: MIT Press“. DOI: 10.1162/tacl_a_00107. Adresse: https://aclanthology.org/Q16-1029/ (Abgerufen am 12. 03. 2025).

[171] A. Yang et al., „Qwen3 Technical Report“, „arXiv:2505.09388 [cs]“, Mai 2025. DOI: 10.48550/arXiv.2505.09388. Adresse: http://arxiv.org/abs/2505.09388 (Abgerufen am 23. 05. 2025).

[172] J. Yang et al., „Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond“, „ACM Trans. Knowl. Discov. Data“, Jg. 18, Nr. 6, 160:1–160:32, Apr. 2024, ISSN: 1556-4681. DOI: 10.1145/3649506. Adresse: https://dl.acm.org/doi/10.1145/3649506 (Abgerufen am 16. 03. 2025).

[173] K. K. Zethsen, „Intralingual translation: an attempt at description“, „Meta: Translators' Journal“, Jg. 54, Nr. 4, S. 795–812, 2009, ISSN: 0026-0452.

[174] J. Zhang und C. Zong, „Neural machine translation: Challenges, progress and future“, en, „Science China Technological Sciences“, Jg. 63, Nr. 10, S. 2028–2050, Okt. 2020, ISSN: 1869-1900. DOI: 10.1007/s11431-020-1632-x. Adresse: https://doi.org/10.1007/s11431-020-1632-x (Abgerufen am 10. 03. 2025).

[175] J. Zhang et al., „Guiding Neural Machine Translation with Retrieved Translation Pieces“, in „Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)“, M. Walker et al., Hrsg., New Orleans, Louisiana: Association for Computational Linguistics, Juni 2018, S. 1325–1335. DOI: 10.18653/v1/N18-1120. Adresse: https://aclanthology.org/N18-1120/ (Abgerufen am 03. 02. 2025).

[176] S. Zhang et al., „Instruction Tuning for Large Language Models: A Survey“, „arXiv:2308.10792 [cs]“, Dez. 2024. DOI: 10.48550/arXiv.2308.10792. Adresse: http://arxiv.org/abs/2308.10792 (Abgerufen am 17. 03. 2025).

[177] T. Zhang et al., „BERTScore: Evaluating Text Generation with BERT“, „ArXiv“, Apr. 2019. Adresse: https://www.semanticscholar.org/paper/BERTScore%3A-Evaluating-Text-Generation-with-BERT-Zhang-Kishore/295065d942abca0711300b2b4c39829551060578 (Abgerufen am 12. 01. 2025).

[178] X. Zhang und M. Lapata, „Sentence Simplification with Deep Reinforcement Learning“, in „Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing“, M. Palmer et al., Hrsg., Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, S. 584–594. DOI: 10.18653/v1/D17-1062. Adresse: https://aclanthology.org/D17-1062/ (Abgerufen am 12. 03. 2025).

[179] Y. Zheng et al., „LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models“, in „Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)“, Y. Cao et al., Hrsg., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, S. 400–410. DOI: 10.18653/v1/2024.acl-demos.38. Adresse: https://aclanthology.org/2024.acl-demos.38/ (Abgerufen am 09. 01. 2025).

[180] H. Zhu et al., „Evaluating Large Language Models for In-Context Learning of Linguistic Patterns In Unseen Low Resource Languages“, in „Proceedings of the First Workshop on Language Models for Low-Resource Languages“, H. Hettiarachchi et al., Hrsg., Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Jan. 2025, S. 414–426. Adresse: https://aclanthology.org/2025.loreslm-1.31/ (Abgerufen am 03. 02. 2025).

[181] W. Zhu et al., „Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis“, „ArXiv“, Jg. abs/2304.04675, 2023. DOI: 10.48550/arXiv.2304.04675. (Abgerufen am 05. 01. 2025).

[182] Z. Zhu et al., „A monolingual tree-based translation model for sentence simplification“, in „Proceedings of the 23rd International Conference on Computational Linguistics“, Ser. COLING ’10, USA: Association for Computational Linguistics, Aug. 2010, S. 1353–1361. (Abgerufen am 12. 03. 2025).

A Anhang

A.1 Verfügbarkeit eines parallelen Korpus je Bundesland

Bundesland	**Paralleler Korpus**
Baden-Württemberg	nein
Bayern	nein
Berlin	nein
Brandenburg	nein
Bremen	nein
Hamburg	ja
Hessen	ja
Mecklenburg-Vorpommern	nein
Niedersachsen	ja
Nordrhein-Westfalen	nein
Rheinland-Pfalz	nein
Saarland	nein
Sachsen	nein
Sachsen-Anhalt	nein
Schleswig-Holstein	nein
Thüringen	nein

Tabelle A.1: Verfügbarkeit eines parallelen Korpus je Bundesland

A.2 Verfügbarkeit eines parallelen Korpus je Landeshauptstadt

Landeshauptstadt	**Paralleler Korpus**
Dresden	ja
Düsseldorf	nein
Erfurt	nein
Hannover	ja
Kiel	nein
Magdeburg	(nein)
Mainz	nein
München	ja
Potsdam	nein
Saarbrücken	ja
Schwerin	nein
Stuttgart	ja
Wiesbaden	nein

Tabelle A.2: Verfügbarkeit eines parallelen Korpus je Landeshauptstadt

A.3 Beispiel des ShareGPT-Formats

```
{
    "conversations": [
        {
            "from": "system",
            "value": "**Rolle:** Du bist ein spezialisierter Assistent für die Übersetzung von Standardsprache in Leichte Sprache in Deutsch.\n\n **Aufgabe:** Wandle den unten angegebenen Text aus der Standardsprache präzise und korrekt in Leichte Sprache um. Beachte dabei strikt die Regeln der Leichten Sprache. Ein Satz in Standardsprache kann dabei zu mehreren Sätzen in Leichter Sprache werden.\n\n **Ausgabeanforderung:**\n **GIB AUSSCHLIESSLICH DEN ÜBERSETZTEN TEXT IN LEICHTER SPRACHE ZURÜCK.**\n **KEINE Einleitungssaetze wie \"Hier ist die Übersetzung:\", KEINE Erklärungen, KEINE Kommentare. Nur der reine Text der Übersetzung.**\n\n **Zu übersetzender Text:**\nDie Folgen des Klimawandels sind weltweit spürbar.\n\n **Erstelle die Übersetzung:**"
        },
        {
            "from": "gpt",
            "value": "Der Klima·wandel hat Folgen. Diese Folgen sind schlecht. Man merkt sie auf der ganzen Welt."
        }
    ]
}
```

Listing A.1: ShareGPT-Format

A.4 Implementierung der Scraper-Template-Klasse

```
import scrapy

class TemplateSpider(scrapy.Spider):
    name = ""
    start_urls = [
        ""
    ]
    allowed_domain = ""

    def parse(self, response):
        article_links = response.css('').getall()
        for link in article_links:
            yield response.follow(link, self.parse_article_page)

    def parse_article_page(self, response):
        # Get the simple version text from the response
        response_text = response.css('').getall()

        # Get the simple version text
        simple_text = ' '.join([
            text.strip()
            for text in response_text
            if text.strip()
        ])

        # Get the normal version link
        normal_link = response.css('').get()

        if normal_link:
            # Create a request for the normal version
            # and pass the simple text data
            yield response.follow(
                normal_link,
                self.parse_normal_page,
                cb_kwargs={
                    'simple_text': simple_text, 'simple_url': response.url
                    }
            )
        else:
            # If no normal version exists, yield only the simple version
            yield {
                'simple_url': response.url,
```

```
43                'simple_text': simple_text,
44                'normal_url': None,
45                'normal_text': None,
46            }
47
48    def parse_normal_page(self, response, simple_text, simple_url):
49        normal_text = ' '.join([
50            text.strip()
51            for text in response.css('').getall()
52            if text.strip()
53        ])
54
55        yield {
56            'simple_url': simple_url,
57            'simple_text': simple_text,
58            'normal_url': response.url,
59            'normal_text': normal_text,
60        }
```

Listing A.2: Scraper-Template

A.5 Implementierung der LLMClient-Klasse

```python
import os

from typing import List, Dict
from openai import OpenAI
from dotenv import load_dotenv, find_dotenv

load_dotenv(find_dotenv)

class LLMClient:

    def __init__(self, model: str):
        self.client = OpenAI(
            base_url="https://openrouter.ai/api/v1",
            api_key=os.getenv("OPENAI_API_KEY")
        )
        self.model = model

    def generate_response(self,
            messages: List[Dict[str, str]],
            temperature: float = 1.0,
            **kwargs):
        content = self.client.beta.chat.completions.parse(
                model = self.model,
                messages = messages,
                temperature = temperature,
                **kwargs
        ).choices[0].message.content

        return content
```

Listing A.3: LLMClient Implementation

A.6 Implementierung der LLMSyntheticDataGenerator-Klasse

```
import json

from typing import List, Dict

class LLMSyntethicDataGenerator:
    SYSTEM_PROMPT = \
    """**Instruction:**
    You are an expert synthetic data generator for German language pairs.

    **Task:**
    Create a short, coherent document about **{topic}** in both standard
    German ("Standardsprache") and Easy German ("Leichte Sprache"). The
    generated document (combined `src` and `tgt` content) should be
    approximately **250-300 tokens** in total.

    The document must consist of approximately **5-10 sentence-aligned
    pairs**. Each pair will have:
    1.  `src`: A sentence in standard German. Sentences can vary in
    complexity.
    2.  `tgt`: The corresponding translation/simplification of the `src`
    sentence into Easy German.

    **Content & Style:**
    *   **Domain:** German public administration (e.g., official notices,
    application processes, regulations).
    *   **Structure:** The document should have a logical flow (e.g., brief
    introduction, main points, brief conclusion).
    *   **Key Principles for `tgt` (Easy German):**
        *   Use very short sentences (one main idea per sentence).
        *   Prefer simple Subject-Verb-Object structure and active voice.
        *  Use present tense or simple past (Praesens/Perfekt). Avoid
    Konjunktiv.
        *   Replace Genitive with "von" + Dative.
        *   Use simple vocabulary. Explain necessary technical terms simply
    on first use.
        *   Break down long compound words (e.g., "Bundes-Gesetz-Blatt").
        *   Spell out abbreviations on first mention (e.g., "zum Beispiel").

    **Example Pair (for illustration of `src`/`tgt` relationship):**
```

```
30      src: "Im Oktober 1920, zur Erinnerung an die 18 gefallenen Soldaten aus
        Dresden-Nickern, im Ersten Weltkrieges fertiggestellt, wurde er nach der
        Zerstoerung Dresdens 1945 fuer die Opfer der Luftangriffe auf Dresden
        umgewidmet."
31      tgt: "Man hat ihn im Oktober 1920 gebaut. Er sollte an 18 Soldaten aus
        Nickern erinnern. Diese Soldaten sind im 1. Weltkrieg gestorben. Nach
        1945 bekam er eine neue Bedeutung. Er erinnert jetzt an die Opfer von
        Bombenangriffen auf Dresden."
32
33      **Output Requirements:**
34      1.  **First, generate the complete document** with the aligned `src` and
        `tgt` sentences as described.
35      2.  **Afterwards, output a JSON object** containing the sentence pairs.
        The JSON should have a single key "pairs", which is a list of objects,
        where each object has an "src" key and a "tgt" key.
36
37      **Generate the document and JSON for the topic: {topic}.**
38      """
39
40      def __init__(self, client: LLMClient):
41          self.client = client
42
43      def generate_data(self,
44              topic: str,
45              limit: int,
46              temperature: float
47      ) -> List[Dict[str, str]]:
48          messages = [
49              {
50                  "role": "system",
51                  "content": self.SYSTEM_PROMPT.format(
52                      limit=limit,
53                      topic=topic
54                  ),
55              },
56          ]
57
58          response_format={
59              "type": "json_schema",
60              "json_schema": {
61                  "name": "synthetic_data_generator",
62                  "strict": True,
63                  "schema": {
64                      "type": "array",
```

```
65                  "description": "An array of sentence pairs, where each
    pair consists of a sentence in German Standardsprache and its
    corresponding translation in German Leichte Sprache.",
66                  "items": {
67                      "type": "object",
68                      "properties": {
69                          "src": {
70                              "type": "string",
71                              "description": "A sentence in standard
    German ('Standardsprache'). The sentence can also be complex to reflect
    occassionaly."
72                          },
73                          "tgt": {
74                              "type": "string",
75                              "description": "The corresponding
    translation/simplification of the `src` sentence into Easy German
    ('Leichte Sprache')"
76                          }
77                      },
78                      "required": [
79                          "src",
80                          "tgt"
81                      ],
82                      "additionalProperties": False
83                  }
84              }
85          }
86      }
87
88      content = self.client.generate_response(
89          messages,
90          temperature = temperature,
91          response_format=response_format
92      )
93      content_json = json.loads(content)
94      return content_json
```

Listing A.4: LLMSyntheticDataGenerator Implementation

A.7 Implementierung der LLMSentenceAligner-Klasse

```
import json

from typing import List, Dict

class LLMSentenceAligner:
    SYSTEM_PROMPT = \
    """**Role:** You are an expert AI system specializing in sentence
    alignment for parallel documents.

    **Task:**
    Align sentences between a source document written in Standard German
    (`source_document`) and its corresponding target document written in
    Easy German (`target_document`). The alignment must map each sentence
    from the `source_document` to one or more sentences in the
    `target_document` that convey the same meaning (1:n alignment).

    The alignment will be based on two provided text blocks:
    1.  `Source document`: The complete text in Standard German.
    2.  `Target document`: The complete text in Easy German, which is a
    simplification/translation of the `Source document`.

    **Process & Requirements:**
    1.  **Sentence Segmentation:** First, accurately segment both the
    `source_document` and `target_document` into individual sentences.
    2.  **Alignment:** For *each* sentence identified in the
    `source_document`, find the corresponding sentence or sequence of
    sentences in the `target_document` that expresses the equivalent
    meaning. Remember that one `source_document` sentence often maps to
    multiple `target_document` sentences due to simplification.

    **Output Requirements:**
    1.  **Content:** The output must *only* consist of the generated
    alignment data.
    2.  **Exclusions:** Do not include any introductory text, explanations,
    or concluding remarks before or after the alignment data. The output
    should be solely the alignment itself.

    **Generate the alignment for the following Source document and Target
    document.**

    Source document:
    {source_document}
```

```
28
29      Target document:
30      {target_document}
31      """
32
33      def __init__(self, client: LLMClient):
34          self.client = client
35
36      def create_alignments_from_documents(self,
37          source_document: str,
38          target_document: str,
39          temperature: float
40      ):
41          messages = [
42              {
43                  "role": "system",
44                  "content":
    self.SYSTEM_PROMPT.format(source_document=source_document,
    target_document=target_document)
45              },
46          ]
47
48          response_format = {
49              "type": "json_schema",
50              "json_schema": {
51                  "name": "document_sentence_alignment_tool",
52                  "strict": True,
53                  "schema": {
54                      "type": "array",
55                      "description": "A list of sentence alignments, where
    each item maps one source sentence to one or more target sentences from
    the respective documents. If there is no corresponding sentence in the
    target document to map to, skip the source sentence.",
56                      "items": {
57                          "type": "object",
58                          "properties": {
59                              "source_sentence": {
60                                  "type": "string",
61                                  "description": "The text of the single full
    sentence from the `source_document`. If there is no corresponding
    sentence in the target document to map to, skip the source sentence or
    provide an empty string ``.  "
62                              },
63                              "target_sentences": {
```

```
64                                  "type": "array",
65                                  "items": {
66                                      "type": "string"
67                                  },
68                                  "description": "A list containing the text
    of the corresponding one or more full sentences from the
    'target_document'. The order of sentences in this list should match
    their order in the 'target_document'. If no corresponding sentence is
    found provide an empty list '[]'."
69                              }
70                          },
71                          "required": [
72                              "source_sentence",
73                              "target_sentences"
74                          ],
75                          "additionalProperties": False
76                      }
77                  }
78              }
79          }
80
81          content = self.client.generate_response(
82              messages,
83              temperature=temperature,
84              response_format=response_format
85          )
86
87          content_json = json.loads(content)
88          return content_json
```

Listing A.5: LLMSentenceAligner Implementation

A.8 Implementierung der HybridSearcher-Klasse

```python
from typing import Dict, List

from qdrant_client import QdrantClient
from fastembed.rerank.cross_encoder import TextCrossEncoder

class HybridSearcher:
    DENSE_MODEL = "jinaai/jina-embeddings-v2-base-de"
    SPARSE_MODEL = "Qdrant/bm25"
    RERANKER_MODEL = 'jinaai/jina-reranker-v2-base-multilingual'

    def __init__(self, client: QdrantClient, collection_name: str):
        self.client = client
        self.collection_name = collection_name

        self.client.set_model(self.DENSE_MODEL)
        self.client.set_sparse_model(self.SPARSE_MODEL)

        self.reranker = TextCrossEncoder(model_name=self.RERANKER_MODEL)

    def _search(self, text: str) -> List[Dict[str, str]]:
        # Perform search and get 10 results
        search_result = self.client.query(
            collection_name=self.collection_name,
            query_text=text,
            query_filter=None,
            limit=10,
        )

        # Extract relevant fields from results
        results = [{
            "source": hit.metadata.get("document"),
            "target": hit.metadata.get("target"),
        } for hit in search_result]

        return results

    def _rerank(self, sentence: str, results: List[Dict[str, str]]) ->
    List[Dict[str, str]]:
        # Extract source texts for reranking
        sources = [result.get("source") for result in results]

        # Perform reranking
```

```
42          new_scores = list(self.reranker.rerank(sentence, sources))
43          ranking = [(i, score) for i, score in enumerate(new_scores)]
44          ranking.sort(key=lambda x: x[1], reverse=True)
45
46          # Return 5 reranked results
47          return [results[rank[0]] for rank in ranking][:5]
48
49      def retrieve_examples(self, sentence: str) -> List[Dict[str, str]]:
50          examples = self._search(text=sentence)
51          reranked_examples = self._rerank(sentence, examples)
52          return reranked_examples
```

Listing A.6: HybridSearcher Implementation

A.9 Implementierung der LLMTranslator-Klasse

```
from typing import List, Dict

class LLMTranslator:
    SYSTEM_PROMPT = \
    """**Rolle:** Du bist ein spezialisierter Assistent für die Übersetzung
    von Standardsprache in Leichte Sprache in Deutsch.

    **Aufgabe:** Wandle den unten angegebenen Text aus der Standardsprache
    präzise und korrekt in Leichte Sprache um. Beachte dabei strikt die
    Regeln der Leichten Sprache. Ein Satz in Standardsprache kann dabei zu
    mehreren Sätzen in Leichter Sprache werden.

    **Ausgabeanforderung:**
    **GIB AUSSCHLIESSLICH DEN ÜBERSETZTEN TEXT IN LEICHTER SPRACHE ZURÜCK.**
    **KEINE Einleitungssätze wie "Hier ist die Übersetzung:", KEINE
    Erklärungen, KEINE Kommentare. Nur der reine Text der Übersetzung.**

    **Zu übersetzender Text:**
    {source}

    **Erstelle die Übersetzung:**
    """

    SYSTEM_PROMPT_RAG = \
    """**Rolle:** Du bist ein spezialisierter Assistent für die Übersetzung
    von Standardsprache in Leichte Sprache (Deutsch).

    **Aufgabe:** Wandle den unten angegebenen Satz aus der Standardsprache
    präzise und korrekt in Leichte Sprache um. Beachte dabei strikt die
    Regeln der Leichten Sprache. Ein Satz in Standardsprache kann dabei zu
    mehreren Sätzen in Leichter Sprache werden.

    **Ausgabeanforderung:**
    **GIB AUSSCHLIESSLICH DEN ÜBERSETZTEN SATZ (BZW. DIE ÜBERSETZTEN SÄTZE)
    IN LEICHTER SPRACHE ZURÜCK.**
    **KEINE Einleitungssätze wie "Hier ist die Übersetzung:", KEINE
    Erklärungen, KEINE Kommentare. Nur der reine Text der Übersetzung.**

    **Beispiele zur Veranschaulichung:**
    Die folgenden Beispiele zeigen, wie die Regeln angewendet werden. Nutze
    sie als Orientierung für Stil und Vereinfachung:
    {formatted_examples}
```

```
31
32      **Zu übersetzender Satz:**
33      {source}
34
35      **Erstelle die Übersetzung:**
36      """
37
38      def __init__(self, client: LLMClient):
39          self.client = client
40
41      def _format_examples(self, examples: List[Dict[str, str]]) -> str:
42          return "\n".join(
43              f"Beispiel {i}:\nStandardsprache: {ex['source']}\nLeichte
    Sprache: {ex['target']}"
44              for i, ex in enumerate(examples, 1)
45          )
46
47      def _create_messages(self, source: str, system_prompt: str, **kwargs) ->
    List[Dict]:
48          return [{
49              "role": "system",
50              "content": system_prompt.format(source=source, **kwargs)
51          }]
52
53      def generate_translation(self, source: str, rag: bool = False, examples:
    List[Dict[str, str]] = None) -> str:
54          if rag:
55              formatted_examples = self._format_examples(examples)
56              messages = self._create_messages(
57                  source,
58                  self.SYSTEM_PROMPT_RAG,
59                  formatted_examples=formatted_examples
60              )
61          else:
62              messages = self._create_messages(
63                  source,
64                  self.SYSTEM_PROMPT
65              )
66          return self.client.generate_response(messages)
```

Listing A.7: LLMTranslator Implementation

A.10 Implementierung der Translator-Klasse

```
from somajo import SoMaJo
from qdrant_client import QdrantClient

class Translator:

    def __init__(self, llm_client: LLMClient, vector_db_client:
    QdrantClient, collection_name: str):
        self.splitter = SoMaJo("de_CMC")
        self.retriever = HybridSearcher(vector_db_client, collection_name)
        self.generator = LLMTranslator(llm_client)

    def translate(self, document: str, rag: bool = False, sentence_mode:
    bool = True):
        if sentence_mode:
            sentences = self.splitter.tokenize_text([document])
            sentence_list = [" ".join([token.text for token in sentence])
    for sentence in sentences]
            translation = ""
            for sentence in sentence_list:
                if rag:
                    examples = self.retriever.retrieve_examples(sentence)
                    translation +=
    f"{self.generator.generate_translation(sentence, rag=rag,
    examples=examples)}\n"
                else:
                    translation +=
    f"{self.generator.generate_translation(sentence)}\n"
        else:
            translation = self.generator.generate_translation(document)
        return translation
```

Listing A.8: Translator Implementation

A.11 Beispielnotebook zum Fine-Tuning des Llama 3.1 8B-Modells mit Unsloth

```
# Run these commands in your notebook first:
# pip install --no-deps bitsandbytes accelerate xformers==0.0.29.post3 peft trl==0.15.2 triton cut_cross_entropy unsloth_zoo
# pip install sentencepiece protobuf "datasets>=3.4.1" huggingface_hub hf_transfer
# pip install --no-deps unsloth

import torch
from datasets import load_dataset
from trl import SFTTrainer, SFTConfig
from unsloth import FastModel
from unsloth.chat_templates import get_chat_template, standardize_data_formats

model, tokenizer = FastModel.from_pretrained(
    model_name="unsloth/Meta-Llama-3.1-8B-Instruct-bnb-4bit",
    max_seq_length=2048,
    load_in_4bit=True,
    dtype=None,
)

model = FastModel.get_peft_model(
    model,
    r=8,
    lora_alpha=8,
    lora_dropout=0,
    bias="none",
    random_state=3407,
    use_rslora=False,
    loftq_config=None,
    finetune_vision_layers=False,
    finetune_language_layers=True,
    finetune_attention_modules=True,
    finetune_mlp_modules=True,
)

tokenizer = get_chat_template(
    tokenizer,
    chat_template="llama-3.1",
)
```

```
38
39 dataset_path = "path/to/your/sharegpt.json"
40 dataset = load_dataset(
41     "json",
42     data_files=dataset_path,
43     split="train",
44 )
45
46 dataset = standardize_data_formats(dataset)
47
48 def apply_chat_template_to_examples(examples):
49     texts = tokenizer.apply_chat_template(
50         examples["conversations"],
51         tokenize=False,
52         add_generation_prompt=False
53     )
54     return {"text": texts}
55
56 dataset = dataset.map(apply_chat_template_to_examples, batched=True)
57
58 trainer = SFTTrainer(
59     model=model,
60     tokenizer=tokenizer,
61     train_dataset=dataset,
62     eval_dataset=None,
63     args=SFTConfig(
64         dataset_text_field="text",
65         max_seq_length=2048,
66         per_device_train_batch_size=2,
67         gradient_accumulation_steps=4,
68         num_train_epochs=2,
69         learning_rate=2e-4,
70         logging_steps=1,
71         optim="adamw_8bit",
72         weight_decay=0.01,
73         lr_scheduler_type="linear",
74         warmup_ratio=0.06,
75         seed=3407,
76         report_to="none",
77         save_strategy="epoch",
78     ),
79 )
80
81 trainer_stats = trainer.train()
```

Listing A.9: Fine-Tuning Notebook

A.12 Detaillierte Evaluationsergebnisse je Konfiguration

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
Llama 3.1 8B	65,163	66,640	0,517	0,541	0,786	0,794	40,673	40,605
Qwen3 8B	64,476	64,980	**0.626**	**0.622**	**0.810**	**0.808**	42,794	42,979
Gemma 3 4B	**68.874**	**70.090**	0,512	0,521	0,778	0,775	**43.714**	**43.258**

Anmerkung: FRE: Flesch-Reading-Ease. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.3: Vergleich der Metriken für alle Modelle der Konfiguration SLM - Dokument mit Durchschnitts- und Medianwerten

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
DeepSeek v3-0324	69,933	72,120	0,492	0,505	**0.754**	**0.750**	47,473	46,725
Gemini 2.5 Flash	**78.199**	**80.630**	**0.528**	**0.519**	0,737	0,737	48,929	48,200
Gemini 2.5 Pro	74,685	73,080	0,466	0,469	0,711	0,706	**49.794**	**49.160**

Anmerkung: FRE: Flesch-Reading-Ease. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.4: Vergleich der Metriken für alle Modelle der Konfiguration SOTA - Dokument mit Durchschnitts- und Medianwerten

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
Llama 3.1 8B	66,340	68,180	0,508	0,532	0,773	0,782	40,833	40,657
Qwen3 8B	64,106	65,675	**0.622**	**0.610**	**0.808**	**0.807**	42,805	42,427
Gemma 3 4B	**74.477**	**73.490**	0,602	0,605	0,776	0,773	**43.972**	**43.633**

Anmerkung: FRE: Flesch-Reading-Ease. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.5: Vergleich der Metriken für alle Modelle der Konfiguration SLM - Satz mit Durchschnitts- und Medianwerten

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
DeepSeek v3-0324	73,413	73,130	**0.546**	**0.546**	**0.748**	**0.748**	48,171	48,220
Gemini 2.5 Flash	**81.211**	**82.000**	0,481	0,484	0,714	0,710	**49.044**	**48.758**

Anmerkung: FRE: Flesch-Reading-Ease. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.6: Vergleich der Metriken für die Konfiguration SOTA - Satz mit Durchschnitts- und Medianwerten

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
Llama 3.1 8B FT	76,173	78,195	0,538	0,548	0,739	0,740	42,531	42,152
Qwen3 8B FT	**77.507**	**79.670**	0,556	0,560	0,745	0,744	**45.210**	**44.720**
Gemma 3 4B FT	76,661	79,565	**0.603**	**0.607**	**0.780**	**0.783**	43,618	42,895

Anmerkung: FRE: Flesch-Reading-Ease; FT: Fine-Tuning. Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.7: Vergleich der Metriken für alle Modelle der Konfiguration SLM - Satz - FT mit Durchschnitts- und Medianwerten

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
Llama 3.1 8B ICL	**73.667**	73,030	0,500	0,523	0,734	0,731	**45.583**	**44.704**
Qwen3 8B ICL	59,677	69,990	**0.651**	**0.662**	**0.807**	**0.810**	44,259	43,960
Gemma 3 4B ICL	73,026	**73.390**	0,569	0,604	0,760	0,770	44,522	43,975

Anmerkung: FRE: Flesch-Reading-Ease; Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.8: Vergleich der Metriken für alle Modelle der Konfiguration SLM - Satz - ICL mit Durchschnitts- und Medianwerten

Modell/Metrik	FRE ↑		BLEU ↑		BERTScore ↑		SARI ↑	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
Llama 3.1 8B FT+ICL	74,412	73,290	0,517	0,536	0,724	0,726	44,984	43,888
Qwen3 8B FT+ICL	**76.239**	**73.795**	0,577	0,577	0,751	0,747	45,492	44,734
Gemma 3 4B FT+ICL	75,404	73,440	**0.602**	**0.609**	**0.765**	**0.765**	**45.669**	**45.059**

Anmerkung: FRE: Flesch-Reading-Ease; FT: Fine-Tuning; Höhere Werte entsprechen bessere Ergebnisse für alle Metriken (↑). Die höchsten Werte je Spalte sind fett markiert.

Tabelle A.9: Vergleich der Metriken für alle Modelle der Konfiguration SLM - Satz - Fine-Tuning und ICL mit Durchschnitts- und Medianwerten

A.13 Textvergleich zum Artikel Euro-WC-Schlüssel

```
Der Euro-WC-Schlüssel ist der Schlüssel für Behindertentoiletten an Autobahnen und in Städten und Gemeinden.
Mit diesem Schlüssel können nicht nur Behindertentoiletten an den deutschen Autobahnen und vielen Städten (auch im europäischen Ausland), sondern auch Einrichtungen wie Aufzüge, Treppenlifte, Rufanlagen und Schranken vor Behindertenparkplätzen benutzt werden.
Auf Initiative der Behindertenbeauftragten der Landeshauptstadt Hannover können Sie den Schlüssel gegen Vorlage des Behindertenausweises für 28.90 Euro (Stand: 2024) in allen Bürgerämtern der Stadt erhalten.
```

Listing A.10: Textvergleich: Input

```
In Deutschland gibt es viele Menschen mit Behinderung.
Und viele von diesen Menschen brauchen eine Behinderten·toilette.
Diese Toiletten können die Menschen mit einem besonderen Schlüssel öffnen.
Dieser Schlüssel heißt: Euro-WC-Schlüssel.
Der Euro-WC-Schlüssel ist ein besonderer Schlüssel.
Mit diesem Schlüssel können Sie zum Beispiel Behinderten·toiletten an Auto·bahnen öffnen.
Oder auch Toiletten in vielen anderen Städten.
Sie sind in einem anderen Land in Europa?
Dann können Sie mit dem Euro-WC-Schlüssel auch in diesen europäischen Ländern die Behinderten·toiletten öffnen.
Mit dem Euro-WC-Schlüssel können Sie auch diese Dinge nutzen: Aufzüge. Treppen·lifte. Ruf·anlagen.
Und Schranken vor Behinderten·park·plätzen.
Nur Menschen mit einem Euro-WC-Schlüssel können die Behinderten·toiletten öffnen.
Wo bekomme ich den Euro-WC-Schlüssel?
Den Euro-WC-Schlüssel können Sie kaufen.
Aber Sie müssen zeigen: Sie haben eine Behinderung.
Sie haben einen Schwer·behinderten·ausweis.
Und der Grad von Ihrer Behinderung muss eine bestimmte Zahl haben.
Nur dann können Sie den Euro-WC-Schlüssel bekommen.
Den Euro-WC-Schlüssel können Sie beim Bürger·amt von Ihrer Stadt kaufen.
Der Euro-WC-Schlüssel kostet 28,90 Euro .
```

Listing A.11: Textvergleich: Referenz

```
Der Euro-WC-Schlüssel ist für Behindertentoiletten.
Er ist für Toiletten an Autobahnen und in Städten und Gemeinden.
Mit dem Schlüssel können Behinderte Toiletten benutzen.
```

```
Der Schlüssel kann auch für Aufzüge, Treppenlifte, Rufanlagen und Schranken
    vor Behindertenparkplätzen verwendet werden.
Die Behindertenbeauftragte der Landeshauptstadt Hannover hat den Schlüssel
    eingeführt.
Der Schlüssel kostet 28,90 Euro.
Sie können den Schlüssel in allen Bürgerämtern der Stadt bekommen.
Dafür müssen Sie den Behindertenausweis vorlegen.
```

Listing A.12: Textvergleich: Qwen3 8B

```
Der Euro-WC-Schlüssel ist ein besonderer Schlüssel.
Er öffnet Toiletten für Menschen mit Behinderungen.
Diesen Schlüssel gibt es bei Autobahnen.
Und auch in Städten und in Dörfern.
Man braucht diesen Schlüssel, um viele Dinge zu benutzen.
Zum Beispiel Behinderten·toiletten an Autobahnen, in vielen Städten.
Auch im europäischen Ausland.
Man kann damit Aufzüge benutzen.
Oder Treppen·lifte.
Man kann auch den Ruf·button nutzen.
Oder Schranken vor Behinderten·Park·plätzen.
Sie können den Schlüssel kaufen.
Gegen Vorlage des Behinderten·ausweises.
Der Schlüssel kostet 28,90 Euro.
Diese Angabe gilt zum Jahr 2024.
Sie erhalten den Schlüssel in allen Bürger·ämtern der Stadt.
Die Behinderten·beauftragte der Landeshauptstadt Hannover hat den Schlüssel
    eingeführt.
```

Listing A.13: Textvergleich: Qwen3 8B FT+ICL

```
Der Euro-WC-Schlüssel ist ein besonderer Schlüssel.
Mit dem Schlüssel kann man Behinderten-Toiletten öffnen.
Diese Toiletten gibt es zum Beispiel an der Autobahn.
Und in Städten.
Und in Gemeinden.
Mit diesem Schlüssel können Sie vieles benutzen.
Zum Beispiel:
Behinderten-Toiletten an der Autobahn in Deutschland.
Behinderten-Toiletten in vielen Städten in Deutschland.
Behinderten-Toiletten auch in anderen Ländern in Europa.
Sie können mit dem Schlüssel auch Aufzüge benutzen.
Oder Treppen-Lifte.
Oder Alarm-Knöpfe.
Oder Schranken zu Parkplätzen für Menschen mit Behinderung.
```

```
Die Beauftragte für Menschen mit Behinderung in Hannover hatte die Idee.
In Hannover gibt es Bürgerämter.
Dort bekommen Sie einen Schlüssel.
Sie müssen Ihren Behinderten-Ausweis zeigen.
Der Schlüssel kostet 28 Euro und 90 Cent.
Der Preis ist von 2024.
```

Listing A.14: Textvergleich: Gemini 2.5 Flash

A.14 Kosten der Evaluation

Modell	Kosten pro Million Token	
	Input ($)	Output ($)
Gemma 3 4B	0,020	0,040
Llama 3.1 8B	0,020	0,030
Qwen3 8B	0,035	0,138
DeepSeek V3-0324	0,300	0,880
Gemini Flash 2.5 Flash	0,150	0,600
Gemini Flash 2.5 Pro	1,250	10,000

Tabelle A.10: API-Kosten ausgewählter LLMs pro Million verarbeiteter Token

Modell	Input	Output Token	Output	Total
	($)	(Anzahl)	($)	($)
Gemma 3 4B	0,001 346 82	34 129	0,001 365 16	0,002 711 98
Llama 3.1 8B	0,001 346 82	33 042	0,000 991 26	0,002 338 08
Qwen3 8B	0,002 356 94	39 277	0,005 420 23	0,007 777 16
DeepSeek V3-0324	0,020 202 30	39 765	0,034 993 20	0,055 195 50
Gemini Flash 2.5 Flash	0,010 101 15	47 418	0,028 450 80	0,038 551 95
Gemini Flash 2.5 Pro	0,084 176 25	417 857	4,178 570 00	4,262 746 25

Tabelle A.11: Kosten auf dokumentbasiertem Input (Prompts: 67341 Token)

Modell	Input	Output Token	Output	Total
	($)	(Anzahl)	($)	($)
Gemma 3 4B	0,006 699 76	45 734	0,001 829 36	0,008 529 12
Llama 3.1 8B	0,006 699 76	33 656	0,001 009 68	0,007 709 44
Qwen3 8B	0,011 724 58	39 593	0,005 463 83	0,017 188 41
DeepSeek V3-0324	0,100 496 40	62 932	0,055 380 16	0,155 876 56
Gemini Flash 2.5 Flash	0,050 248 20	62 379	0,037 427 40	0,087 675 60

Tabelle A.12: Kosten auf satzbasiertem Input (Prompts: 334988 Token)

Modell	Input	Output Token	Output	Total
	($)	(Anzahl)	($)	($)
Gemma 3 4B	0,012 468 60	45 734	0,001 829 36	0,014 297 96
Llama 3.1 8B	0,012 468 60	33 656	0,001 009 68	0,013 478 28
Qwen3 8B	0,021 820 05	39 593	0,005 463 83	0,027 283 88
DeepSeek V3-0324	0,187 029 00	62 932	0,055 380 16	0,242 409 16
Gemini Flash 2.5 Flash	0,093 514 50	62 379	0,037 427 40	0,130 941 90

Tabelle A.13: Kosten auf satzbasiertem Input mit ICL (Prompts: 623430 Token)

A.15 Verwendete Hilfsmittel

In der Tabelle A.14 sind die im Rahmen der Bearbeitung des Themas der Masterarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tool	Verwendung
Overleaf	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments
Gemini 2.5 Pro	Erstellung und Formatierung von Tabellen und Formatierung von Text sowie Rechtschreib- und Grammatikprüfung
Zotero	Speicherung, Verwaltung und Zitation von Literatur für dieses Dokument

Tabelle A.14: Verwendete Hilfsmittel und Werkzeuge

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