THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Bachelor Thesis

Konzeption und Implementierung einer barrierefreien mobilen Anwendung zur Förderung der kognitiven und sprachlichen Zugänglichkeit durch KI-gestützte Textvereinfachung

zur Erlangung des akademischen Grades

Bachelor of Science

eingereicht im Fachbereich Mathematik, Naturwissenschaften und Informatik an der Technischen Hochschule Mittelhessen

von

Silke Poelmann

5. August 2025

Referent: Kevin Linne

Korreferent: Prof. Dr. Dennis Priefer

Erklärung zur Verwendung von generativer KI

In Übereinstimmung mit den Empfehlungen der Deutschen Forschungsgemeinschaft (DFG)[1] und denen der Zeitschrift Theoretical Computer Science[2] erkläre ich (der Autor/die Autorin) hiermit den Einsatz von generativer KI.

Bei der Vorbereitung dieser Arbeit habe ich ChatGPT 4 verwendet, um ausschließlich die Lesbarkeit und Sprache zu verbessern. Nach der Verwendung von ChatGPT 4 habe ich den Inhalt überprüft und nach Bedarf bearbeitet und übernehme die volle Verantwortung für den Inhalt dieser Arbeit.

Gender-Disclaimer

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Gießen, den 5. August 2025 Silke Poelmann

1 DFG formuliert Richtlinien für den Umgang mit generativen Modellen für Text- und Bild: https://www.dfg.de/resource/blob/289676/89c03e7a7a8a0240936029959748 32f9/230921-statement-executive-committee-ki-ai-data.pdf

2 Erklärung zur Verwendung von generativer KI in wissenschaftlichen Arbeiten: https://www.sciencedirect.com/journal/theoretical-computer-science/publish/guide-for-authors

Diese Bachelorarbeit untersucht, wie Künstliche Intelligenz zur automatisierten Vereinfachung komplexer Texte in Leichter Sprache beitragen kann, um den Zugang zu Informationen für Menschen mit eingeschränkter Sprachkompetenz zu verbessern. Zentrale Fragestellung ist, wie eine barrierefreie Anwendung gestaltet werden kann, die nutzerfreundlich ist und sprachliche Hürden abbaut. Dazu werden verschiedene Transformer-Modelle (FLAN-T5, mT5, LeoLM – bereits finegetunt im Projekt *Erlesen*) in Bezug auf Prompting-Strategien evaluiert. FLAN-T5 und mT5 werden zusätzlich mittels Fine-Tuning auf deutschsprachige, regelkonforme Datensätze angepasst. Auf dieser Grundlage wird ein Modell ausgewählt und in einen funktionalen Prototyp integriert, der als Progressive Web App umgesetzt ist und dabei grundlegende Anforderungen an Barrierefreiheit (WCAG 2.2) sowie die Regeln der Leichten Sprache berücksichtigt. Im Entwicklungsprozess zeigen sich Herausforderungen bei Datenverfügbarkeit, Modellqualität und Rechenleistung. Insgesamt bestätigt die Arbeit die technische Umsetzbarkeit des Ansatzes und zeigt zugleich Potenziale für Weiterentwicklungen – etwa durch satzweise Vereinfachung, synthetische Daten und ein lernfähiges System mit Nutzerfeedback. So wird ein praxisnaher Beitrag zur Förderung digitaler Inklusion geleistet.

Inhaltsverzeichnis

1 Einführung

Schriftliche Sprache ist ein zentrales Mittel der Kommunikation und spielt in der heutigen Gesellschaft eine wichtige Rolle, sei es in der Schule, im Beruf oder im privaten Alltag. Texte begegnen uns überall. Sie informieren, strukturieren Abläufe und unterstützen uns bei Entscheidungen, zum Beispiel durch Hinweisschilder, Gebrauchsanleitungen, E-Mails oder offizielle Schreiben. Mit der fortschreitenden Digitalisierung verlagert sich ein großer Teil der schriftlichen Kommunikation in den digitalen Raum, etwa über Webseiten, digitale Anwendungen oder soziale Medien. Dadurch eröffnen sich neue Kanäle der Informationsvermittlung, gleichzeitig steigen jedoch die Anforderungen, diese Inhalte zu finden, zu verstehen und zu verarbeiten.

1.1 Problembeschreibung und Motivation

Lese- und Schreibkompetenz sind keine Selbstverständlichkeiten, sondern Fähigkeiten, die bei Menschen unterschiedlich stark ausgeprägt sind. Dabei geht es nicht nur darum, Buchstaben zu entziffern oder Wörter zu schreiben, sondern auch darum, Inhalte im jeweiligen Kontext zu verstehen, zu bewerten und anzuwenden. Diese Fähigkeiten werden unter dem Begriff *Literalität* zusammengefasst [Bud20].

Laut der LEO-2018-Studie haben rund 6,2 Millionen Erwachsene in Deutschland erhebliche Schwierigkeiten beim Lesen, Schreiben und Verstehen von Texten. Besonders herausfordernd sind für sie alltägliche Situationen wie das Lesen von Beipackzetteln, das Ausfüllen von Formularen, das Verstehen schriftlicher Arbeitsanweisungen oder der digitale Fahrkartenkauf [Gro19]. Dabei ist zu beachten, dass die tatsächliche Zahl deutlich höher liegt, da viele weitere betroffene Personengruppen in der Studie nicht berücksichtigt wurden.

Um diesen Menschen den Zugang zu Informationen zu erleichtern, wurde das Konzept der ***Leichten Sprache*** entwickelt. Ziel ist es, komplexe Inhalte so zu vereinfachen, dass sie auch von Menschen mit geringer Literalität verstanden werden können [Net22].

Durch gesetzliche Regelungen wie die Barrierefreie-Informationstechnik-Verordnung (BITV 2.0) [Bun11] ist Leichte Sprache im öffentlichen Raum deutlich sichtbarer geworden. Besonders auf den Webseiten von Behörden und öffentlichen Einrichtungen kommt sie vermehrt zum Einsatz – etwa bei der Bundeszentrale für politische Bildung (bpb)[1] oder dem Bundesministerium für Arbeit und Soziales (BMAS)[2].

Seit dem Inkrafttreten des Barrierefreiheitsstärkungsgesetzes (BFSG) [Bun21] gewinnt die barrierefreie Gestaltung digitaler Angebote auch im privaten Sektor an Bedeutung. Viele Maßnahmen orientieren sich an den WCAG-Kriterien [Coo18], die vor allem technische und visuelle Hürden adressieren. Sprachliche Zugangshürden bleiben hingegen weitgehend unberücksichtigt. Zwar gibt es einzelne Angebote in Leichter Sprache, etwa auf Webseiten oder in digitalen Anwendungen, diese sind jedoch meist auf bestimmte Inhalte beschränkt. Betroffene können oft nicht selbst entscheiden, welche Informationen sie vereinfacht benötigen. Eine flexible Lösung zur individuellen Nutzung komplexer Inhalte fehlt bislang.

In diesem Zusammenhang rückt der Einsatz **Künstlicher Intelligenz (KI)** zunehmend in den Fokus. KI-gestützte Sprachverarbeitung eröffnet neue Möglichkeiten, komplexe Texte automatisiert zu analysieren, zu vereinfachen und zielgruppengerecht aufzubereiten. Erste Anwendungen zur Textvereinfachung sind bereits verfügbar[3], richten sich jedoch meist an Unternehmen und sind kostenpflichtig. Für viele Betroffene sind diese Lösungen im Alltag kaum nutzbar, insbesondere bei privaten Nachrichten oder spontan benötigten Informationen.

Die Entscheidung, sich im Rahmen dieser Bachelorarbeit mit Leichter Sprache und **digitaler Barrierefreiheit** zu befassen, folgt der Motivation, sprachliche Teilhabe im privaten Alltag zu stärken. Menschen mit geringen sprachlichen Kompetenzen sind bisher stark eingeschränkt, wenn es darum geht, sich eigenständig zu informieren, zu kommunizieren oder an gesellschaftlichen Themen teilzunehmen. Doch nur wer etwa einen Zeitungsartikel zu politischen Ereignissen versteht, kann Entwicklungen einordnen, sich eine fundierte Meinung bilden und aktiv am öffentlichen Diskurs mitwirken.

1 bpb: Informationen in Leichter Sprache. https://www.bpb.de/die-bpb/ueber-uns/informationen-in-leichter-sprache/, (Zugriff am 15.04.2025).
2 BMAS: Leichte Sprache. https://www.bmas.de/DE/Leichte-Sprache/leichte-sprache.html, (Zugriff am 15.04.2025].
3 Summ.ai: KI-gestützte Textvereinfachung. https://summ-ai.com/, (Zugriff am 15.04.2025).

1.2 Ziele der Arbeit

Ziel dieser Arbeit ist es, die Zugänglichkeit von Informationen für benachteiligte Nutzergruppen zu verbessern und einen Beitrag zur digitalen Inklusion zu leisten. Dazu wird eine zentrale Forschungsfrage formuliert, die durch zwei untergeordnete Fragestellungen ergänzt wird. Diese werden im Rahmen der Konzeption und prototypischen Umsetzung systematisch untersucht:

F1 (Hauptforschungsfrage)

Wie kann eine barrierefreie mobile Anwendung entwickelt werden, die mithilfe KI-gestützter Textvereinfachung in Leichter Sprache den Zugang zu Informationen erleichtert und das Textverständnis verbessert?

F1.1

Wie lassen sich die Regeln der Leichten Sprache in einen automatisierten KI-Prozess zur Textvereinfachung überführen?

F1.2

Welche Anforderungen sind bei der Entwicklung einer mobilen Anwendung besonders wichtig, um eine benutzerfreundliche und zugängliche Nutzung für die Zielgruppe sicherzustellen?

1.3 Vorgehensweise

Das methodische Vorgehen dieser Arbeit umfasst vier aufeinander aufbauende Phasen.

In der **ersten Phase** wird durch eine umfassende Literatur- und Quellenrecherche das theoretische Fundament gelegt. Ziel ist es, zentrale Grundlagen für das Themenverständnis und die weitere Bearbeitung zu schaffen. Im Fokus stehen unter anderem folgende Fragen:

- Wer zählt zur Zielgruppe, und welche sprachlichen Hürden bestehen?
- Was ist Leichte Sprache, wie ist sie entstanden und gesetzlich verankert?
- Welche Regelwerke gibt es und wie bewertet man Leichte Sprache?
- Welche Möglichkeiten bietet KI zur Textvereinfachung, und welche Modelle stehen zur Verfügung?
- Welche KI-Ansätze existieren bereits, und welche Probleme zeigen sich?

In der **zweiten Phase** erfolgt zunächst eine Anforderungsanalyse, in der funktionale und nicht-funktionale Anforderungen – einschließlich barrierebezogener Aspekte – definiert werden. Darauf aufbauend wird ein Designkonzept entwickelt, das sich an den Bedürfnissen der Zielgruppe orientiert. Im Anschluss werden Auswahlkriterien für ein geeignetes KI-Modell zur Textvereinfachung festgelegt. Ein Beispieltext wird mit verschiedenen Sprachmodellen vereinfacht, die Ergebnisse anhand der Kriterien bewertet. Auf dieser Grundlage erfolgt die Auswahl eines geeigneten Modells – ggf. inklusive Datenaufbereitung und Fine-Tuning mit regelkonformen Datensätzen zur Anpassung an Leichte Sprache. Abschließend werden Systemarchitektur und technische Grundlagen der Anwendung definiert.

In der **dritten Phase** wird ein funktionaler Prototyp entwickelt, der das zuvor erarbeitete Konzept technisch umsetzt. Der Schwerpunkt liegt auf der Integration der gewählten KI-Lösung sowie auf einer barrierefreien und zielgruppengerechten Gestaltung der Benutzeroberfläche.

In der **vierten und letzten Phase** werden die Ergebnisse zusammengefasst und reflektiert. Zunächst beleuchtet die Diskussion methodische Herausforderungen und inhaltliche Einschränkungen. Darauf folgt das Fazit, das zentrale Erkenntnisse im Hinblick auf die Forschungsfragen zusammenfasst. Abschließend gibt ein Ausblick Hinweise auf mögliche Weiterentwicklungen und zukünftige Forschungsperspektiven.

1.4 Abgrenzung

Im Rahmen dieser Arbeit werden bewusst bestimmte Aspekte ausgeklammert, um den Fokus auf die KI-basierte Umsetzung Leichter Sprache zu legen. Das Konzept der Leichten Sprache sowie die zugehörigen Regeln werden dabei nicht kritisch hinterfragt, sondern dienen als verbindliche Grundlage für die Umsetzung. Die mobile Anwendung orientiert sich an den Bedürfnissen der Zielgruppe. Die Richtlinien zur digitalen Barrierefreiheit (WCAG) fließen in die Konzeption ein und werden gestalterisch berücksichtigt. Eine vollständige Barrierefreiheitsprüfung sowie die Erstellung einer offiziellen Erklärung sind jedoch nicht vorgesehen. Zudem wird kein eigenes Sprachmodell entwickelt. Stattdessen kommt ein bestehendes KI-Modell zum Einsatz, das bei Bedarf angepasst wird. Auch rechtliche Aspekte im Zusammenhang mit dem KI-Einsatz werden lediglich theoretisch eingeordnet. Die Verständlichkeit der generierten Texte wird nicht durch Nutzergruppen getestet, sondern ausschließlich anhand etablierter Lesbarkeitskriterien bewertet.

1.5 Struktur der Arbeit

Diese Arbeit gliedert sich in fünf Kapitel:

- **Kapitel 1 – Einführung:** Dieses Kapitel enthält die Problembeschreibung und Motivation, formuliert die Ziele der Arbeit, erläutert das methodische Vorgehen, grenzt den Untersuchungsrahmen ab und stellt die Struktur der Arbeit dar.
- **Kapitel 2 – Hintergrund:** Hier werden die theoretischen Grundlagen erarbeitet. Dazu zählen die Zielgruppe und ihre sprachlichen Herausforderungen, zentrale Aspekte der Leichten Sprache sowie Grundlagen der KI-gestützten Textvereinfachung.
- **Kapitel 3 – Konzept:** In diesem Kapitel wird die konzeptionelle Entwicklung der mobilen Webanwendung beschrieben. Es werden Anforderungen an Technik, Design und Inhalt definiert, geeignete Technologien und Modelle ausgewählt und erste Entwürfe für die Benutzeroberfläche erstellt.
- **Kapitel 4 – Implementierung:** Dieses Kapitel widmet sich der praktischen Umsetzung. Ein funktionsfähiger Prototyp wird entwickelt, wobei der Fokus auf der technischen Implementierung und der barrierefreien Gestaltung der Anwendung liegt.
- **Kapitel 5 – Zusammenfassung:** Zum Abschluss werden die zentralen Ergebnisse der Arbeit zusammengefasst, kritisch reflektiert und im Ausblick mögliche Weiterentwicklungen aufgezeigt.

Diese Struktur gewährleistet, dass alle Forschungsfragen systematisch bearbeitet und die Zielsetzung der Arbeit umfassend erfüllt werden können.

2 Hintergrund

2.1 Zielgruppen und ihre sprachlichen Herausforderungen

Dieses Kapitel beschreibt, für wen Leichte Sprache besonders wichtig ist und welche sprachlichen Barrieren beim Lesen und Verstehen typischerweise auftreten.

2.1.1 Primäre und sekundäre Zielgruppe

Leichte Sprache ermöglicht Menschen mit Lese- und Verständnisschwierigkeiten einen gleichberechtigten Zugang zu Informationen. In Deutschland betrifft dies schätzungsweise rund 20 Millionen Menschen (vgl. Anhang A.1). Auf Grundlage verschiedener Regelwerke [Inc17, Net22], Fachliteratur [Maa16] und gesetzlicher Vorgaben [Bun02] lassen sich drei Typen sprachlicher Hürden unterscheiden[1], die häufig kombiniert auftreten:

- **Kognitive Verarbeitungshürden:** Schwierigkeiten beim Denken, Verstehen oder Erinnern (z. B. bei geistiger Behinderung oder Demenz).
- **Sprachliche Verständnishürden:** Probleme beim Verstehen sprachlicher Strukturen wie Wortbedeutungen, Satzbau oder Grammatik (z. B. nach einem Schlaganfall oder bei geringer Deutschkompetenz).
- **Lesebezogene Dekodierhürden:** Einschränkungen beim Erkennen, Entziffern und Verstehen geschriebener Wörter und Sätze (z. B. bei Lese-Rechtschreib-Schwäche oder fehlender Leseerfahrung).

> Diese Arbeit richtet sich **primär** an Personen mit den oben genannten Typen sprachlicher Hürden, die im Alltag auf verständliche Texte angewiesen sind. **Sekundär** richtet sie sich an Anbieter, die freiwillig barrierearme Inhalte erstellen, insbesondere Betreiber privater Webseiten mit begrenzten finanziellen Mitteln.

1 Diese Einteilung dient der Orientierung und ist nicht abschließend.

2.1.2 Sprachliche und visuelle Barrieren

Für die beschriebenen Zielgruppen entstehen insbesondere dann Verständnisschwierigkeiten, wenn Texte bestimmte sprachliche oder visuelle Merkmale aufweisen. Die folgende Tabelle fasst typische Barrieren zusammen, die in verschiedenen Regelwerken übereinstimmend genannt werden [Bun11, Net22, Inc17].

Tabelle 2.1: Typische Kommunikationsbarrieren bei Texten

Ebene		**Barrieren entstehen ...**
Wort	1.	wenn Fachbegriffe und Fremdwörter verwendet werden.
	2.	wenn Abkürzungen verwendet werden.
	3.	wenn unterschiedliche Wörter für denselben Inhalt verwendet werden.
	4.	wenn die Wörter zu lang sind.
Satz	5.	wenn die Sätze zu lang und verschachtelt sind.
	6.	wenn die Sätze zu viele Informationen enthalten.
	7.	wenn schwierige grammatische Formen verwendet werden (Verneinungen, Passiv, Konjunktiv oder Genitiv).
Struktur	8.	wenn der Text schlecht gegliedert ist (fehlende Überschriften, unklare Struktur).
	9.	wenn die Absätze zu lang sind.
	10.	wenn wichtige Informationen nicht hervorgehoben oder an das Ende gestellt werden.
Layout	11.	wenn die Schrift schwer lesbar ist (Größe, Kontrast, Schriftart).
	12.	wenn die Textanordnung verwirrend ist (Ausrichtung, Zeilenumbrüche, Silbentrennung).
	13.	wenn visuelle Hilfsmittel fehlen oder das Layout überladen ist (keine Bilder/Hervorhebungen, zu viele Elemente/Farben).

Die Anwendung soll dazu beitragen, diese Barrieren abzubauen und das Textverständnis zu erleichtern.

2.2 Leichte Sprache

Dieses Kapitel behandelt das Konzept der Leichten Sprache als zentrales Mittel zur Verbesserung sprachlicher Zugänglichkeit. Es beginnt mit einer Begriffsbestimmung und Abgrenzung zur Einfachen Sprache, gefolgt von einem historischen Überblick und den rechtlichen Rahmenbedingungen. Daran anschließend werden zentrale Regelwerke vorgestellt sowie Verfahren zur automatisierten Bewertung erläutert. Abschließend erfolgt ein Blick auf die Umsetzung in der Praxis.

2.2.1 Begriffsdefinition und Abgrenzung

Leichte Sprache ist eine bewusst vereinfachte Form der deutschen Standardsprache mit dem Ziel, Texte besser verständlich zu machen. Davon abzugrenzen ist die Einfache Sprache. Beide unterscheiden sich in mehreren zentralen Merkmalen.

Leichte Sprache

- Feste, verbindliche Regeln
- Einfachster Sprachstil
- Sehr kurze Sätze, nur Hauptsätze
- Nur einfachste Begriffe
- Strenge Layout-Regeln
- Inhalte stark reduziert
- Maximal vereinfachtes Deutsch
- Primäre Zielgruppe (vgl. 2.1.1)

[Kel14, Maa16]

Einfache Sprache

- Flexible Regeln
- Komplexerer Sprachstil
- Längere Sätze, Nebensätze erlaubt
- Alltägliche Begriffe
- Flexibles Layout
- Inhalte vollständig erhalten
- Allgemeinverständliches Deutsch
- Breite Bevölkerung

[Kel14, Neu25]

Hauptunterschied:
Leichte Sprache folgt festen, verbindlichen Regeln gegenüber der Standardsprache, die frei gestaltet werden kann. Einfache Sprache bewegt sich flexibel in der Mitte dazwischen [Maa16].

Leichte Sprache **Einfache Sprache** **Standardsprache**

2.2.2 Historische Entwicklung in Deutschland

Seit den 1960er-Jahren rückten Teilhabe und verständliche Kommunikation für Menschen mit Behinderungen stärker in den Fokus. Erste Selbstvertretungsbewegungen entstanden. In Schweden entstand in dieser Zeit das Konzept ***Lättläst***, das Menschen mit kognitiven oder sprachlichen Einschränkungen den Zugang zu verständlicher Literatur erleichtern sollte [Reg13, S. 64]. Es gilt als früher Meilenstein sprachlicher Vereinfachung und dient seitdem auch anderen Ländern als Orientierung [Reg13, S. 63].

In Deutschland gewann Leichte Sprache ab den 1990er-Jahren an Bedeutung. Die US-amerikanische Selbstvertretungsbewegung People First brachte Ideen von Selbstbestimmung nach Deutschland. Menschen mit Lernschwierigkeiten organisierten sich erstmals selbst für ihre Rechte [Mena]. Ein Schlüsselereignis in diesem Zusammenhang war der **Kongress der Bundesvereinigung Lebenshilfe** 1994 in Duisburg. Dort machten Betroffene deutlich, wie wichtig sprachliche Barrierefreiheit für gleichberechtigte Teilhabe ist [Net21]. Der Mitorganisator Martin Th. Hahn beschreibt die Veranstaltung als Auslöser eines grundlegenden Wandels, der die Teilhabe aller Menschen in den Mittelpunkt rückt [HahoJ, S. 2-3.].

Mit dem Projekt *Wir vertreten uns selbst!* (1997–2001) begann die deutsche **People-First**-Bewegung und legte erste Grundlagen für Leichte Sprache. Daraus gingen ein eigener Verein **Mensch zuerst** – Netzwerk People First Deutschland e.V. (2001), zwei Wörterbücher (2000, 2008) und eine Petition zur gesetzlichen Verankerung (2009) hervor. 2006 wurde das **Netzwerk Leichte Sprache** gegründet – ein Zusammenschluss von Menschen mit und ohne Lernschwierigkeiten –, das den Einsatz Leichter Sprache in Alltag, Politik und Verwaltung fördert [Netc, Netb]. 2009 veröffentlichte das Netzwerk ein Regelwerk, das bis heute als zentrale Orientierung für barrierearme Texte gilt [Maa16, S.17] (vgl. Abschnitt 2.2.4).

Auch auf europäischer Ebene entstand 2009 mit dem Regelwerk von **Inclusion Europe** eine wichtige Grundlage für verständliche Kommunikation. Es gilt als bewährte Praxis in der EU [Inc17, Alv23] (vgl. Abschnitt 2.2.4).

2.2.3 Gesetzliche Verankerung in Deutschland

1994 markierte einen rechtlichen Wendepunkt: Mit der Aufnahme des Satzes *„Niemand darf wegen seiner Behinderung benachteiligt werden“* in **Artikel 3 des Grundgesetzes** [Bun49] wurde die Grundlage für barrierefreie Kommunikation in Deutschland gelegt.

Leichte Sprache im Öffentlichen Sektor

Mit Inkrafttreten der **UN-Behindertenrechtskonvention (UN-BRK)** im Jahr 2009 wurde Leichte Sprache ausdrücklich als Bestandteil barrierefreier Kommunikation anerkannt [Uni06]. Um die darin formulierten Vorgaben europaweit einheitlich umzusetzen, verabschiedete die Europäische Union im Jahr 2016 die **Richtlinie 2016/2102**. Diese verpflichtet öffentliche Stellen dazu, ihre Websites und mobilen Anwendungen barrierefrei zugänglich zu gestalten [Eur16]. In Deutschland erfolgte die Umsetzung der Richtlinie durch die Anpassung des **Behindertengleichstellungsgesetzes (BGG)** im Jahr 2018. Besonders relevant ist hierbei insbesondere **§ 11 BGG**, der öffentliche Stellen verpflichtet, auf Verlangen Informationen auch in Leichter Sprache bereitzustellen [Bun02].

Ergänzend wurde die **Barrierefreie-Informationstechnik-Verordnung (BITV 2.0)** angepasst, um konkrete Anforderungen an die digitale Barrierefreiheit zu definieren [Bun11] (vgl. Abschnitt 2.2.4). Grundlage hierfür bildet Artikel 3 der Verordnung, der auf die europäische Norm **DIN EN 301 549**[1] verweist. Diese Norm legt detaillierte Anforderungen an barrierefreie Informations- und Kommunikationstechnologien fest und nimmt Bezug auf die **Web Content Accessibility Guidelines (WCAG 2.1)**, die als international anerkannter Standard für digitale Barrierefreiheit gelten [Coo18] (vgl. Abschnitt 2.2.4).

Leichte Sprache im privaten Sektor

Im privaten Bereich bestehen bislang keine ausdrücklichen gesetzlichen Verpflichtungen zur Verwendung Leichter Sprache. Dennoch lassen sich auch hier wichtige Entwicklungen auf die **UN-Behindertenrechtskonvention (UN-BRK)** zurückführen. Mit der **EU-Richtlinie 2019/882 (European Accessibility Act)** wurde der Geltungsbereich auf private Anbieter ausgeweitet [Eur19]. In Deutschland wurde die Richtlinie durch das **Barrierefreiheitsstärkungsgesetz (BFSG)** umgesetzt, das ab Juni 2025 gilt. Es verpflichtet Anbieter und Hersteller, etwa in den Bereichen Verkehr, Telekommunikation oder E-Commerce, zur barrierefreien Gestaltung ihrer Produkte und Dienstleistungen. Leichte Sprache wird zwar nicht explizit genannt, doch betont § 1 BFSG die Teilhabe von Menschen mit Behinderungen, was auch kognitive Zugänglichkeit einschließt [Bun21]. Technisch verweist das BFSG – wie auch die BITV 2.0 – auf die europäische Norm **DIN EN 301 549** und damit auch auf die **WCAG 2.1**. Letztere fordern zwar keine Leichte Sprache, betonen jedoch im Rahmen des Prinzips der Verständlichkeit die Bedeutung klarer und einfacher Sprache [Coo18] (vgl. Abschnitt 2.2.4).

2.2.4 Regelwerke und Standards der Leichten Sprache

In Deutschland existieren verschiedene Regelwerke zur Leichten Sprache – aus Praxis, Normung und Gesetzgebung –, die sich in Herkunft, Zielsetzung und Methodik unterscheiden.

Inclusion Europe: Das europäische Regelwerk von Inclusion Europe entstand 2009 im Projekt *Pathways* und wurde in mehr als 15 Sprachen übersetzt. Es umfasst 132 Regeln in sechs Kategorien (vgl. Anhang A.2), die medienübergreifend auf eine verständliche Aufbereitung von Informationen für Menschen mit Lernschwierigkeiten abzielen. Es gilt europaweit als grundlegende Orientierungshilfe [Inc17, Alv23].

1 DIN EN 301 549, online unter: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf, abgerufen am 13. Mai 2025.

Netzwerk Leichte Sprache: Das Regelwerk des Netzwerks Leichte Sprache wurde 2009 veröffentlicht und 2022 überarbeitet. Es entstand in inklusiver Zusammenarbeit von Prüfenden und Übersetzenden und richtet sich an beide Gruppen. Die 47 Regeln sind in sechs Kategorien gegliedert (vgl. Anhang A.2) und bilden in Deutschland den meistgenutzten Standard [Net22, Maa15]. Drei Gründe tragen dazu bei: Erstens stimmen seine Inhalte weitgehend mit der BITV 2.0 überein, auch wenn es dort nicht direkt genannt wird (vgl. Anhang A.3). Zweitens wurde es 2013 gemeinsam mit dem Bundesministerium für Arbeit und Soziales speziell für die öffentliche Verwaltung neu aufgelegt [Net14]. Drittens wirkte das Netzwerk an der Erarbeitung der DIN SPEC 33429 mit [DIN25].

BITV 2.0: Die Barrierefreie-Informationstechnik-Verordnung (BITV 2.0) ist seit 2011 die zentrale Rechtsgrundlage für digitale Barrierefreiheit und wurde zuletzt 2023 aktualisiert. Seit 2018 verpflichtet sie öffentliche Stellen des Bundes[1], ihre digitalen Angebote barrierefrei zu gestalten. Laut § 4 BITV 2.0 müssen bestimmte Informationen auf der Startseite zusätzlich in Leichter Sprache bereitgestellt werden – darunter eine Beschreibung der Hauptinhalte, Hinweise zur Navigation und zur Barrierefreiheit. Die Ausgestaltung ist in Anlage 2 geregelt, die 13 Anforderungen zur sprachlichen und visuellen Gestaltung benennt [Bun11] (vgl. Anhang A.2) .

WCAG: Die vom World Wide Web Consortium (W3C) entwickelten Web Content Accessibility Guidelines (WCAG) richten sich an Entwickler und Designer. Obwohl im Oktober 2023 die aktuelle Fassung **WCAG 2.2** erschienen ist [Cam23], ist in Deutschland bislang noch die WCAG 2.1 über die BITV 2.0 verbindlich [Bun11] . Ziel ist es, digitale Inhalte möglichst vielen Menschen zugänglich zu machen – unter anderem für Menschen mit Seh-, Hör-, motorischen oder kognitiven Einschränkungen, ältere Personen und Menschen mit geringen Deutschkenntnissen [dIufH24]. Die WCAG beruhen auf vier Prinzipien: Inhalte sollen (1) wahrnehmbar, (2) bedienbar, (3) verständlich und (4) robust sein. Daraus leiten sich 13 Zielsetzungen mit 86 Erfolgskriterien ab, die den Konformitätsstufen A, AA und AAA zugeordnet sind. Zwar fordern die WCAG keine Leichte Sprache, doch kann sie ein hilfreiches Mittel sein, um das Prinzip der Verständlichkeit zu erfüllen. Relevant sind insbesondere die Richtlinien 3.1.3 bis 3.1.6 zu Wortwahl, Abkürzungen, Leseniveau und Aussprache [Cam23].

1 Vgl. § 1a BGG. Dazu zählen z. B. Bundesbehörden, Körperschaften und Anstalten des öffentlichen Rechts. Eine vollständige Aufzählung findet sich unter: https://www.gesetze-im-internet.de/bgg/, abgerufen am 21. Mai 2025.

DIN SPEC 33429: Im März 2025 wurde mit der DIN SPEC 33429 erstmals eine umfassende Norm zur Leichten Sprache in Deutschland veröffentlicht. Sie entstand unter Beteiligung des Bundesministeriums für Arbeit und Soziales, des Netzwerks Leichte Sprache sowie weiterer Fachstellen. Die Norm ist rechtlich nicht bindend, dient jedoch als praxisorientierte Empfehlung für alle, die Texte in Leichter Sprache erstellen oder in Auftrag geben. Sie formuliert einheitliche Standards zur Sicherung der Textqualität und zum Abbau sprachlicher Barrieren, insbesondere für Menschen mit Lernschwierigkeiten. Neben sprachlichen und visuellen Vorgaben enthält sie medienübergreifende Empfehlungen sowie Anforderungen an Erstellung und Qualifikation [DIN25]. Sie bündelt bestehende Ansätze und bietet einen orientierenden Qualitätsrahmen für künftige Anwendungen [fAuS25].

2.2.5 Automatisierte Bewertungsverfahren für Leichte Sprache

Die folgende Tabelle zeigt verschiedene Verfahren zur Bewertung von Leichter Sprache und ordnet ein, ob diese automatisiert durchgeführt werden können.

Tabelle 2.2: Metriken zur Messung von Leichter Sprache

Verfahren	**Grundlage / Methode**	**Automatisierbar**
Lesbarkeitsformeln (quantitativ)	Basierend auf Wort-/ Satzlänge, Silbenzahl und Wortkomplexität: • Flesch (Deutsch) • Lix Lesbarkeits-Index • SMOG-Index (Deutsch) • Wiener Sachtextformel	Ja
Toolgestützte Analyse (quantitativ)	Kombinierte Analysetools (KI, Regeln und Metriken): • Wortliga (zugänglich) • capito (zugänglich) • Optimeil Prüfer (zugänglich) • Flesch-Reading-Ease (zugänglich)	Ja
Regelbasierte Prüfung (quantitativ/qualitativ)	Prüfung anhand von Regelwerken nach Wort-, Satz-, Text- und Layoutmerkmalen: • BITV 2.0 • Inclusion Europe • Netzwerk Leichte Sprache • DIN SPEC 33429	Teilweise
Expertenbeurteilung (qualitativ)	Manuelle Bewertung durch geschulte Personen, meist aus der Zielgruppe selbst	Nein

Lesbarkeitsformeln wie Flesch (Deutsch), LIX, SMOG oder die Wiener Sachtextformel ermöglichen eine erste Einschätzung der Verständlichkeit. Sie basieren auf quantifizierbaren Merkmalen wie Wortlänge, Satzlänge und Silbenanzahl und lassen sich automatisiert berechnen [TUE16].

Regelbasierte Verfahren wie die DIN SPEC 33429 definieren Anforderungen an Wortwahl, Satzbau, Textstruktur und Layout, die teilweise automatisiert überprüft werden können [DIN25].

Frei zugängliche Tools wie Wortliga [Wor] und capito [cap] kombinieren Lesbarkeitsmetriken mit regelbasierten und teilweise KI-gestützten Verfahren. Die zugrunde liegenden Bewertungsansätze variieren zum Teil erheblich und sind nicht durchgängig transparent dokumentiert.

Eine abschließende qualitative Bewertung durch Personen aus der Zielgruppe kann zusätzliche Hinweise zur Verständlichkeit liefern [DIN25].

2.2.6 Praktische Anwendung Leichter Sprache

In der Praxis wird Leichte Sprache bislang nur punktuell eingesetzt. Institutionen wie das Bundesministerium für Arbeit und Soziales[1], Aktion Mensch[2] oder die Lebenshilfe[3] bieten zwar Inhalte in Leichter Sprache an, jedoch meist nur ergänzend und zu ausgewählten Themen. Vollständige Parallelfassungen zur Standardsprache sind bislang selten. Gleichzeitig wächst das Angebot Einfacher Sprache, etwa durch die Tagesschau in Einfacher Sprache[4]. Diese erfüllen jedoch nicht die Regeln Leichter Sprache und erreichen daher nur Teile der Zielgruppe.

Auch in der Selbstvertretung bestehen strukturelle Hürden: Während das Netzwerk Leichte Sprache weiterhin aktiv ist [Neta], mussten andere Organisationen wie „Mensch zuerst“ ihre Arbeit mangels Förderung einschränken [Menb].

Mit der DIN SPEC 33429 liegt seit 2025 erstmals ein normativer Orientierungsrahmen für Leichte Sprache vor [DIN25]. Diese Entwicklung unterstreicht das wachsende Interesse an verbindlichen Standards. Gleichzeitig gewinnt der Einsatz von KI-gestützten Vereinfachungstools an Bedeutung. Näheres dazu findet sich in Abschnitt 2.3.7.

1 https://www.bmas.de/DE/Leichte-Sprache/leichte-sprache.html (Zugriff am 20.06.2025)
2 https://www.aktion-mensch.de/leichte-sprache.html (Zugriff am 20.06.2025)
3 https://www.lebenshilfe.de/leichte-sprache (Zugriff am 20.06.2025)
4 https://www.tagesschau.de/multimedia/sendung/tagesschau_in_einfacher_sprache (Zugriff am 20.06.2025)

2.3 Künstliche Intelligenz und Textvereinfachung

Im digitalen Alltag wächst die Informationsflut stetig. Viele Inhalte sind sprachlich komplex und für die in dieser Arbeit betrachtete Zielgruppe schwer zugänglich. **Künstliche Intelligenz (KI)** kann helfen, Texte zu vereinfachen und so sprachliche Barrieren abzubauen. Dieses Kapitel erläutert zentrale Grundlagen der KI mit Fokus auf die Verarbeitung natürlicher Sprache (Natural Language Processing, **NLP**). Anschließend wird die Textvereinfachung als konkrete NLP-Anwendung vorgestellt. Darauf folgen geeignete Modelle, Trainingsverfahren und relevante rechtliche Rahmenbedingungen.

2.3.1 Grundlagen und Methoden Künstlicher Intelligenz

Definition und Funktionsweise von KI-Systemen

Diese Arbeit orientiert sich an der KI-Definition der Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (OECD) aus dem Jahr 2024. Die OECD ist ein Zusammenschluss von 38 Mitgliedsländern [OEC24], darunter Deutschland. Ihre international anerkannte Definition von Künstlicher Intelligenz bildet auch die Grundlage der EU-KI-Verordnung [Eur24], die in Abschnitt 2.3.6 näher erläutert wird. Der Begriff Künstliche Intelligenz wird im Folgenden sinngemäß wiedergegeben.

Ein KI System ist ein computerbasiertes System zur Erreichung eines bestimmten Ziels. Es verarbeitet Eingaben wie Texte, Bilder oder Sprache, analysiert sie mit einem internen Modell und erzeugt entsprechende Ausgaben wie Vorhersagen, Inhalte oder Entscheidungen. Einige Systeme sind lernfähig und verbessern sich fortlaufend durch neue Daten oder Rückmeldungen [OEC24].

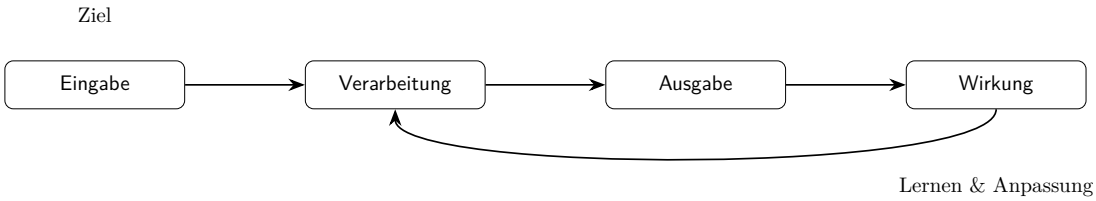


Abbildung 2.1: Funktionsweise eines KI-Systems

Übertragen auf den Kontext dieser Arbeit bedeutet das: Ziel des KI-Systems ist die automatisierte Vereinfachung von Texten. Es erhält komplexe Texte als Eingabe, verarbeitet sie modellbasiert und gibt vereinfachte Texte als Ausgabe zurück. Diese werden für Nutzer besser zugänglich. Zudem kann das System durch Nutzerfeedback dazulernen und seine Leistung verbessern.

Relevante Methoden und Anwendungsbereiche der KI

Innerhalb der Künstlichen Intelligenz lassen sich methodische und anwendungsbezogene Teilbereiche unterscheiden. Für diese Arbeit sind insbesondere folgende drei relevant:

- **Maschinelles Lernen (Machine Learning, ML):**
  Typ: Methode | Teil von: KI
  Systeme erkennen Muster in Daten und lernen daraus selbstständig – ohne dass jeder Schritt programmiert sein muss [Paa20].
  Beispiele: Produktempfehlungen, Spamfilter

- **Tiefes Lernen (Deep Learning, DL):**
  Typ: Methode | Teil von: ML
  Künstliche neuronale Netze mit vielen Schichten erfassen komplexe Muster in großen Datenmengen [Paa20].
  Beispiele: Bild- und Spracherkennung, GPT

- **Verarbeitung natürlicher Sprache (Natural Language Processing, NLP):**
  Typ: Anwendung | nutzt: ML und DL
  Systeme analysieren und erzeugen Sprache mithilfe lernbasierter Verfahren [Paa20].
  Beispiele: Textvereinfachung [Dad21], Übersetzungen, Chatbots

Die folgende Abbildung zeigt die Einordnung dieser Bereiche und ihre Überschneidungen:

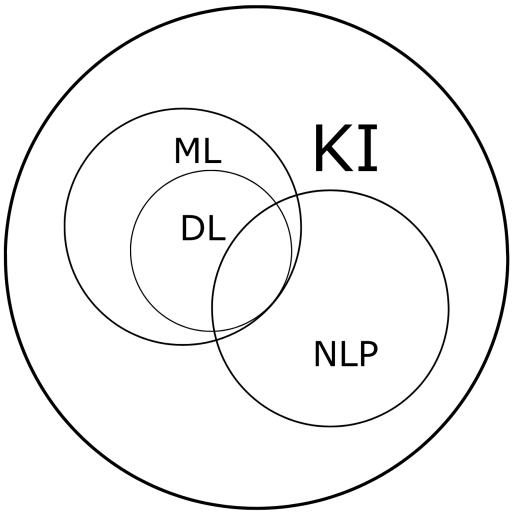


Abbildung 2.2: Darstellung der Beziehungen zwischen ML, DL und NLP.

2.3.2 Natural Language Processing (NLP)

Anwendungsbereiche des NLP

Natural Language Processing (NLP) ist ein zentraler Anwendungsbereich der Künstlichen Intelligenz und beschäftigt sich mit der automatisierten Verarbeitung natürlicher Sprache in Text- oder Sprachform. Ziel ist die Umwandlung unstrukturierter Spracheingaben in strukturierte Daten zur anschließenden Analyse und semantischen Auswertung, sodass Systeme sprachliche Informationen verstehen und darauf reagieren können.

Zur Umsetzung nutzt NLP Methoden der Künstlichen Intelligenz, insbesondere **Machine Learning (ML)**. ML-Modelle erkennen Muster in Sprachdaten und lernen anhand von Beispielen (überwachtes Lernen), selbstständig (unüberwachtes Lernen) oder durch Vorhersage (selbstüberwachtes Lernen). So können Modelle beispielsweise einfache und komplexe Sätze unterscheiden. **Deep Learning (DL)** setzt dabei mehrschichtige neuronale Netze ein, die Daten zunehmend abstrakt verarbeiten und tiefere Sprachstrukturen erfassen [Paa20].

NLP umfasst zwei Hauptaspekte: das **Natural Language Understanding (NLU)** für das Verstehen und Interpretieren, sowie das **Natural Language Generation (NLG)** für die Erzeugung natürlicher Sprache aus strukturierten Daten [Ans24].

Wichtige Anwendungsbereiche sind [Ans24]:

- **Sentimentanalyse:**
  Erkennung von Gefühlen und Meinungen in Texten.
- **Falschinformations-Erkennung:**
  Identifikation von Fake News, Spam oder Propaganda.
- **Maschinelle Übersetzung:**
  Automatische Übersetzung zwischen Sprachen.
- **Frage-Antwort-Systeme:**
  Automatische Beantwortung von Fragen.
- **Textzusammenfassung:**
  Erstellung kurzer Zusammenfassungen langer Texte.
- **Textvereinfachung:**
  Umformulierung komplexer Texte in leicht verständliche Sprache [Dad21].

NLP-Workflow

Der systematische Ablauf des Natural Language Processing (NLP) gliedert sich in vier zentrale Schritte [Tau22, S.121-125], [Gri19]:

Schritt 1: Eingabe
Der Rohtext wird dem System als Eingabe zugeführt. Liegt die Information in anderer Form vor, zum Beispiel als Audiodatei, wird sie zuerst per automatischer Spracherkennung (Speech-to-Text) in Text umgewandelt.

Schritt 2: Bereinigung und Vorverarbeitung
Der Text wird anschließend bereinigt und vorverarbeitet. Dazu zählen unter anderem die Zerlegung in einzelne Einheiten wie Wörter oder Symbole (Tokenisierung), die Vereinheitlichung der Schreibweise durch Kleinschreibung und Entfernung von Satzzeichen (Normalisierung) sowie die Aufteilung in Sätze oder Absätze (Segmentierung). Diese Schritte bereiten die Daten für die weitere Analyse vor.

Schritt 3: Sprachverständnis und Verarbeitung
Dies ist meist der komplexeste und rechenintensivste Schritt. Er umfasst die Analyse sprachlicher Strukturen (z. B. Part-of-Speech-Tagging, Parsing), die inhaltliche Bedeutungsanalyse (z. B. semantische Analyse, Named Entity Recognition, Sentimentanalyse) sowie anwendungsspezifische Verfahren wie Informationsgewinnung oder automatische Textzusammenfassung. Hier gewinnt das System verwertbare Erkenntnisse.

Schritt 4: Ausgabe
Die Ergebnisse liegen nun in strukturierter Form vor, etwa als Daten, Zusammenfassungen oder Extraktionen. Je nach Anwendung erfolgt die Ausgabe als Text oder wird per Text-to-Speech wieder in gesprochene Sprache umgewandelt.

Dieser Workflow lässt sich grafisch wie folgt veranschaulichen:

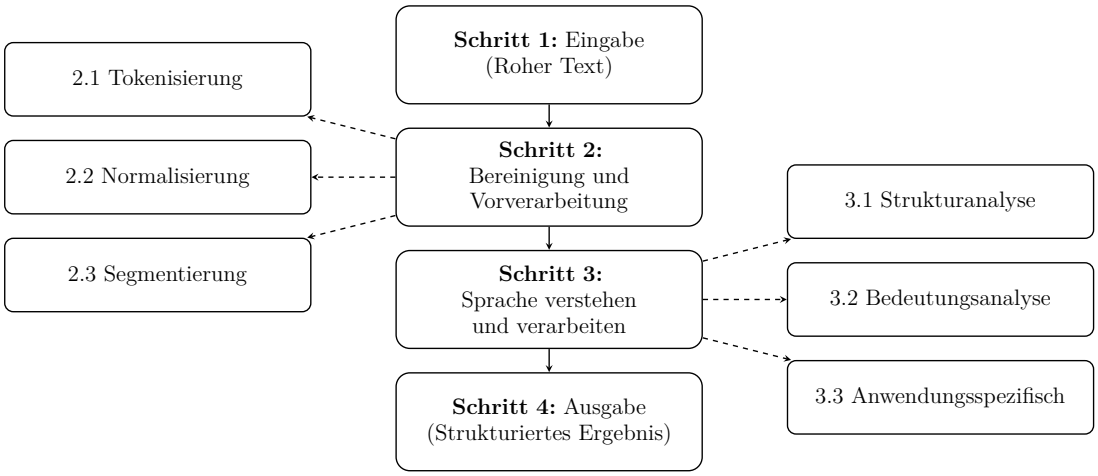


Abbildung 2.3: Überblick des NLP-Workflows mit Hauptschritten und Unterprozessen

Herausforderungen bei der automatischen Sprachverarbeitung

Die automatische Verarbeitung natürlicher Sprache stellt Computersysteme vor zahlreiche Herausforderungen. Beispielsweise ist Sprache oft mehrdeutig und Begriffe können je nach Kontext unterschiedliche Bedeutungen haben. Die Sprache unterliegt einem ständigen Wandel, der die Verarbeitung erschwert, da laufend neue Ausdrücke durch Trends, Jugendsprache und Anglizismen entstehen[1]. Zudem machen Menschen Grammatikfehler oder verwenden unvollständige Sätze. Besonders schwierig ist es, emotionale Nuancen wie Ironie oder Sarkasmus korrekt zu interpretieren, wenn Kontext oder nonverbale Hinweise fehlen, die für das Verständnis notwendig sind [Tau22, S.119-120], [Gri19].

2.3.3 Textvereinfachung (Text Simplification, TS) als Anwendungsfeld

Die **Textvereinfachung (engl. Text Simplification, TS)** ist ein Teilbereich des NLP. Ihr Ziel ist es, Texte automatisch so umzuschreiben, dass sie leichter lesbar und besser verständlich werden, ohne wesentliche Informationen zu verlieren [Dad21]. Dafür werden verschiedene Bearbeitungsschritte, sogenannte Transformationen, auf Satzebene angewendet. Zu den wichtigsten gehören [AM20]:

- **Substitution:** Ersetzen komplexer Wörter/Phrasen durch einfachere Synonyme
- **Deletion:** Entfernen unnötiger Satzteile oder Informationen
- **Reordering:** Umstellung der Satzstruktur zur besseren Verständlichkeit
- **Splitting:** Aufteilen komplexer Sätze in mehrere einfachere Sätze

Moderne Ansätze zur Textvereinfachung sind überwiegend datengetrieben und basieren auf überwachtem Lernen mit **parallelen Korpora**. Dabei handelt es sich um große Sammlungen von Textpaaren aus Original- und vereinfachten Texten, bei denen jeder komplexe Satz mit seiner vereinfachten Version verknüpft ist. Durch diese paarweise Ausrichtung lernen die Modelle die Transformationen automatisch [AM20]. Diese satzbasierten Ansätze stoßen jedoch bei längeren Dokumenten an ihre Grenzen [Aum22]. Moderne Ansätze kombinieren daher Textvereinfachung mit **Zusammenfassungstechniken**, um sowohl sprachlich als auch inhaltlich zugängliche Texte zu erstellen.

1 2024 wurden 3000 neue Wörter in den Duden aufgenommen. https://www.duden.de/duden-2024 (Zugriff am 22.06.2025)

2.3.4 Modelle zur Textvereinfachung

Im Rahmen dieser Arbeit liegt der Fokus ausschließlich auf modernen neuronalen Modellansätzen, insbesondere auf der Transformer-Architektur, da diese den aktuellen Stand der Technik in der automatischen Textvereinfachung repräsentiert [AM20, Vas23].

Transformer-Modelle zeichnen sich durch drei zentrale Eigenschaften aus: Erstens nutzen sie den sogenannten **Self-Attention-Mechanismus**, der es dem Modell ermöglicht, bei jedem Verarbeitungsschritt gezielt auf alle relevanten Teile des Textes zu achten. So können selbst in langen und komplexen Sätzen inhaltliche Zusammenhänge zuverlässig erkannt werden, das macht Transformer-Modelle besonders leistungsfähig bei anspruchsvollen Aufgaben wie der Textvereinfachung. Zweitens erlaubt die Architektur eine vollständig **parallele Verarbeitung** der Eingabedaten. Im Gegensatz zu früheren Ansätzen mit sequentieller Verarbeitung, bei denen Wörter nacheinander bearbeitet werden mussten, können Transformer-Modelle viele Wörter gleichzeitig verarbeiten. Dadurch sind sie deutlich schneller beim Training und in der Anwendung. Drittens sind Transformer-Modelle hochgradig **skalierbar**: Sie lassen sich effizient auf sehr große Datenmengen anwenden und ermöglichen Modelle mit Milliarden von Parametern. Das macht sie besonders effizient im Einsatz bei datenintensiven Aufgaben [Vas23].

Architekturvarianten und ihre Anwendung

Transformer-Modelle lassen sich grundsätzlich in drei Hauptarchitekturen unterteilen [Fu23]:

- **Encoder-only-Modelle (Textverständnis)**:
  Eigenschaft: Diese Modelle verarbeiten Texte bidirektional (in beide Richtungen) und sind darauf spezialisiert, Texte zu analysieren und zu verstehen, ohne selbst neue Inhalte zu generieren [Dev19].
  Beispiele: BERT, RoBERTa, DistilBERT, DeBERTa-v3 [Ans24]

- **Decoder-only-Modelle (Textgenerierung)**:
  Eigenschaft: Diese Modelle verarbeiten Text von links nach rechts und generieren dabei sequenziell neue Wörter basierend auf dem bisherigen Kontext [Rad18].
  Beispiele: GPT-4, Llama 2/3, Mistral 7B [Ans24]

- **Encoder-Decoder-Modelle (Sequence-to-Sequence)**:
  Eigenschaft: Diese Modelle kombinieren beide Ansätze mit einem separaten Encoder für das Textverständnis und einem Decoder für die Textgenerierung [Vas23].
  Beispiele: T5, Flan-T5, BART, mT5, PEGASUS [Ans24]

Training und Evaluierung

Für die Anwendung von Transformer-Modellen auf die Textvereinfachung stehen verschiedene Trainingsansätze zur Verfügung, die sich in Aufwand und Ressourcenbedarf unterscheiden:

Pre-Training (Vortraining): Pre-Training bezeichnet das vollständige Training eines Sprachmodells ohne Vorwissen (*from scratch*) auf großen, unannotierten Textkorpora. Dabei lernt das Modell grundlegende sprachliche Muster, Grammatik und Semantik durch unüberwachtes Lernen. Solche Modelle werden als vortrainierte Sprachmodelle oder, bei besonders großen Varianten, als **Large Language Models (LLMs)** bezeichnet. Das Vortraining ist rechen- und speicherintensiv und verursacht hohe Kosten. Für die Textvereinfachung allein reicht Pre-Training nicht aus, da keine spezifischen Vereinfachungsregeln erlernt werden. Eine weitere Anpassung ist daher erforderlich [Ans24, Paa20].

Fine-Tuning (Feinanpassung): Fine-Tuning nutzt vortrainierte LLMs und passt sie durch zusätzliches Training an eine spezifische Aufgabe wie die Textvereinfachung an. Dabei wird das vorhandene Sprachverständnis des Modells genutzt, um es gezielt auf neue Anforderungen auszurichten. Im Gegensatz zum Pre-Training erfolgt das Training mit annotierten Beispieldaten durch überwachtes Lernen, meist in Form paralleler Korpora (Eingabe-Ausgabe-Paare). Fine-Tuning ist weniger rechenintensiv und benötigt deutlich kleinere Datenmengen [Ans24, Paa20].

Prompting (Eingabeaufforderung): Prompting nutzt vortrainierte LLMs gezielt für spezifische Aufgaben wie die Textvereinfachung. Im Unterschied zum Fine-Tuning ist keine nachträgliche Anpassung des Modells nötig, was den Rechenaufwand deutlich verringert. Stattdessen erhält das Modell eine strukturierte Eingabe, einen sogenannten *Prompt*.

Die **drei häufigsten Prompt-Strategien** sind [Ans24]:

1. Anweisungsbasiertes Lernen (*Instruction-based Learning*)

Das Modell erhält eine klare Aufgabe und ggf. Beispiele.

Eingabe: „Vereinfache in Leichter Sprache:
Obwohl das Kraftfahrzeug alt ist, wird es noch genutzt."
Ausgabe: „Das Auto ist alt. Aber man benutzt es noch."

2. Vorlagenbasiertes Lernen (*Template-based Learning*)

Das Modell nutzt eine strukturierte Vorlage mit Lücken oder Auswahl.

Eingabe: „Ergänze das fehlende Wort in Leichter Sprache:
Das _____ hat vier Räder."
Ausgabe: „Das Auto hat vier Räder."

3. Stellvertreteraufgaben (*Proxy-Task-based Learning*)

Eine komplexe Aufgabe wird in eine einfachere verwandelt.

Originalaufgabe: „Vereinfache in Leichter Sprache."
Proxy-Task-Prompt: „Erkläre den Text mit einfachen Worten."
Eingabe: „Das Kraftfahrzeug wurde abgeschleppt."
Ausgabe: „Ein Auto stand falsch. Ein anderes Auto hat es mitgenommen."

Zur Bewertung der Textvereinfachung werden vorwiegend BLEU und **SARI** eingesetzt. BLEU alleine eignet sich nicht für Leichte Sprache, da es Ausgaben bevorzugt, die dem Original ähnlich sind. Leichte Sprache verwendet dagegen oft kürzere, aber zahlreichere Sätze. SARI kombiniert BLEU-Komponenten und bewertet gezielt Vereinfachungen wie **Hinzufügen, Beibehalten** und **Entfernen** von Textteilen [Säu20]. Weitere spezifische Metriken für Leichte Sprache sind in Abschnitt 2.2.5 beschrieben.

2.3.5 Parallele Korpora für das Training von Sprachmodellen

Wie im vorherigen Abschnitt erläutert, basiert das Fine-Tuning vortrainierter Sprachmodelle für die Textvereinfachung auf parallelen Korpora mit Eingabe-Ausgabe-Paaren. Diese bestehen aus Satzpaaren in Standardsprache und vereinfachter Sprache. Dabei handelt es sich nicht immer um eine 1:1-Zuordnung. Häufig stehen einem Standardsatz mehrere vereinfachte Sätze gegenüber (1:n-Zuordnung), um die sprachliche Komplexität zu reduzieren [Mad23, AM20].

Erstellungsmethoden

Zur Erstellung paralleler Datensätze stehen drei Verfahren zur Verfügung: **Manuell** erfolgt die Übersetzung durch Experten und liefert qualitativ hochwertige Ergebnisse, ist jedoch mit erheblichem Zeit- und Kostenaufwand verbunden. **Automatisch** können vorhandene Standard- und Einfachtexte, etwa durch Crawling von Behörden-Webseiten, gesammelt werden; die automatisierte Zuordnung von Satzpaaren ist dabei jedoch oft fehleranfällig. Eine **synthetisch** Methode nutzt Sprachmodelle, um vereinfachte Texte zurück in Standardsprache zu übersetzen und künstliche Satzpaare zu erzeugen [Klö24].

Herausforderungen

Die Erstellung paralleler Korpora für die Textvereinfachung ist mit besonderen Herausforderungen verbunden:

- **Keine Satzpaare:** Häufig fehlen direkte Zuordnungen zwischen Standard- und Leichter Sprache. Stattdessen bestehen die Korpora aus lediglich thematisch ähnlichen Texten [Mad23].
- **Uneinheitliche Standards:** Vor der Einführung der DIN-Norm (vgl. Abschnitt 2.2.4) wurden verschiedene Regelwerke verwendet, die sich in Aufbau und Umfang voneinander unterschieden (Anhang A.2). Dies führt zu inkonsistenten Trainingsgrundlagen.
- **Unklare Datenherkunft:** Bei vielen Korpora ist nicht dokumentiert, ob urheberrechtlich geschützte oder personenbezogene Inhalte enthalten sind. Solche Datensätze gelten als *Black Box* und werfen rechtliche und ethische Fragen auf.
- **Vermischung von Sprachstufen:** Oft fehlt eine klare Trennung zwischen Leichter und Einfacher Sprache, obwohl beide unterschiedliche Zielgruppen und Regeln haben.

2.3.6 Rechtliche Aspekte beim Einsatz von KI

Für den produktiven Einsatz einer mobilen KI-Anwendung zur Textvereinfachung bestehen wichtige rechtliche Fragestellungen. Im Fokus stehen dabei zwei zentrale Bereiche: die eingehenden Daten (**Input**) und die erzeugten Ergebnisse (**Output**). Beim **Input** ist die Herkunft der Daten vortrainierter Modelle oder Fine-Tuning-Datensätze häufig unklar. Dies kann zu Urheberrechtsverletzungen (UrhG[1]) und Datenschutzverstößen (DSGVO[2]) führen, insbesondere wenn personenbezogene oder geschützte Inhalte betroffen sind. Auch Nutzereingaben bergen vergleichbare Risiken und sollten durch klare Nutzungsbedingungen abgesichert werden. Beim **Output** können automatisch erzeugte Texte fehlerhaft oder missverständlich sein. Dies ist besonders bei sensiblen Inhalten wie juristischen oder medizinischen Texten problematisch und wirft Haftungsfragen auf. Seit August 2024 gilt in Deutschland der **EU AI Act**, das erste umfassende KI-Gesetz weltweit. Für diese Anwendung bedeutet das, dass Nutzer ab August 2026 darüber informiert werden müssen, dass sie mit einem KI-System interagieren [Eur24].

1 Vgl. UrhG online, https://www.gesetze-im-internet.de/urhg (Zugriff am 24.06.2025)
2 Vgl. DSGVO online, https://dsgvo-gesetz.de/ (Zugriff am 24.06.2025)

2.3.7 Stand der Technik und Forschung

In Deutschland gibt es verschiedene KI-Ansätze, die standardsprachliche Texte in Leichte Sprache umwandeln. Das zeigt das wachsende Bewusstsein für barrierefreie Kommunikation in Praxis und Forschung. Im Folgenden werden ausgewählte kommerzielle Tools und Forschungsprojekte vorgestellt sowie zentrale Herausforderungen hinsichtlich Barrierefreiheit und Zielgruppenorientierung diskutiert.

Kommerzielle Anwendungen

Ein Teil der aktuellen Entwicklungen erfolgt im kommerziellen Bereich. Im Folgenden werden drei KI-gestützte Tools vorgestellt, um Unterschiede in den verwendeten Sprachmodellen, den methodischen Grundlagen sowie den Preismodellen herauszuarbeiten.

Tabelle 2.3: Kommerzielle KI-Tools für Leichte Sprache – Vergleichsübersicht

Kategorie	**Leichte Sprache Übersetzer**	**capito**	**Wortliga**
KI-Modell	OpenAI	Kombination führender LLMs	ChatGPT (GPT-4)
Methodische Grundlagen	keine Angaben	capito-Kriterienkatalog (160 Regeln) vom Netzwerk Leichte Sprache	Übersetzung gemäß DIN 8581-1
Preismodell	kostenlos (5.400 Zeichen/Monat), kostenpflichtiges Abo-Modell	kostenlos (3.600 Zeichen/Monat), kostenpflichtiges Abo-Modell	kostenlos (1.000 Zeichen pro Text), kostenpflichtige Lizenzen (Monat/Jahr/Lifetime)
Besonderheit	–	Gütesiegel, TÜV-Zertifizierung	Schwerpunkt auf Einfacher Sprache
Quellen	[Hal25]	[CFS25]	[WOR25]

Die drei kommerziellen Tools basieren auf großen Sprachmodellen: Sowohl der Leichte-Sprache-Übersetzer [Hal25] als auch Wortliga [WOR25] setzen auf GPT, während Capito [CFS25] eine Kombination mehrerer LLMs verwendet. Zu den methodischen Grundlagen, also der spezifischen Trainingsweise für Leichte Sprache, machen die Anbieter keine detaillierten Angaben. Bekannt sind jedoch die Orientierungsrahmen: Capito folgt dem Regelwerk des Netzwerks Leichte Sprache [CFS25], Wortliga wirbt mit DIN-konformer Übersetzung [WOR25]. Alle Tools bieten ähnliche Preismodelle mit kostenlosen Basisversionen sowie erweiterten, kostenpflichtigen Varianten.

Forschungsprojekte

Neben kommerziellen Entwicklungen widmen sich auch Forschungsprojekte dem Thema. Im Folgenden werden vier ausgewählte Vorhaben hinsichtlich technischer Umsetzung und wissenschaftlicher Ausrichtung analysiert.

Tabelle 2.4: KI-Projekte für Leichte Sprache – Vergleichsübersicht

Kategorie	**KI-GesKom**	**ErLeSen**	**LeiSA**	**STARK-LS**
Institution/ Träger	Uni Hildesheim	FH Aachen	DABB / KISZ	HKA Karlsruhe / JGU Mainz
Projektzeitraum	abgeschlossen	abgeschlossen	aktiv (bis ?)	aktiv (bis 2029)
Forschungs-bereich	Interlinguale Übersetzung	Leichte Sprache – Erstellung und Analyse	Vereinfachung behördlicher Texte	Inklusion am Arbeitsplatz
Textdomäne	Gesundheit	Allgemeine Texte	Verwaltungs- und Behördentexte	Arbeitsmaterialien
KI-Modell	SUMM AI (LLM-basiert)	GPT-2, GPT-2-xl, Leo-7B, Leo-13B	LLaMa 3	SUMM AI (LLM-basiert)
Methodische Grundlagen	170 Gesundheitstexte (Apotheken Umschau)	8.130 Dokumentpaare, Plan-basierte Steuerung	Satzpaare aus Verwaltungstexten, Few-Shot-Learning und Fine-Tuning	nicht spezifiziert
Evaluation	quantitativ und qualitativ	quantitativ und qualitativ	quantitativ und qualitativ	quantitativ und qualitativ
Open Source	nein (kommerziell SUMM AI)	ja	ja	nicht spezifiziert
Quellen	[Uni25]	[FH 25, Klö24]	[Dig25]	[Hoc25]

Die Übersicht zeigt: Die aktuellen Forschungsprojekte zur Leichten Sprache sind praxisnaher geworden. Statt allgemeiner Machbarkeitsstudien geht es mehr um konkrete Anwendungen, zum Beispiel in Verwaltung, Gesundheit und Arbeitswelt. Alle Projekte nutzen vorhandene Technologien. Dabei kommen sowohl Open-Source-Modelle wie GPT, LeoLM [FH 25] und LLaMa [Dig25] als auch kommerzielle Systeme wie SUMM AI [Uni25, Hoc25] zum Einsatz. Neue Modelle werden meist nicht entwickelt. Stattdessen liegt der Fokus darauf, bestehende Sprachmodelle gezielt anzupassen, anzuwenden und zu bewerten. Die Bewertung erfolgt systematisch, mit quantitativen und qualitativen Methoden, meist unter realistischen Bedingungen. Die Tabelle zeigt damit deutlich: Die Forschung bewegt sich immer mehr in Richtung praxisnahe und einsatzbereite Lösungen.

Einschränkungen und Herausforderungen

Spezialisierte KI-Tools zur Vereinfachung von Texten in Leichter Sprache wie Wortliga, der Leichte-Sprache-Übersetzer (beide basieren auf GPT) und Capito (nutzt mehrere große Sprachmodelle) zeigen ein typisches Muster: Sie setzen vor allem formale Regeln zuverlässig um. Dazu zählen kurze Sätze, eine einfache Gliederung sowie der Verzicht auf bildhafte Sprache oder Metaphern. Diese oberflächlichen Anpassungen gelingen technisch gut. Deutlich schwieriger sind jedoch sprachlich-inhaltliche Anforderungen. Viele Systeme verwenden weiterhin komplexe Nominalstil-Konstruktionen, schwierige Begriffe werden kaum oder gar nicht erklärt, und vor allem geht häufig der ursprüngliche Sinn des Textes verloren. Die vereinfachten Versionen entsprechen zwar oft formal den Vorgaben, geben die Inhalte aber nur verkürzt oder verzerrt wieder (vgl. Anhang A.4 und A.5).

Die Projekte *LeiSA* und *Erlesen*, die ebenfalls auf Decoder-Only-Modellen basieren, zeigen ähnliche Probleme auf: Die eingesetzten Modelle verzichten entweder vollständig auf erklärende Beispiele oder vereinfachen Texte so stark, dass wesentliche Inhalte verloren gehen und nicht mehr nachvollziehbar sind [Rad25, Klö24].

Auch auf der Anwendungsebene bestehen wesentliche Barrieren: Viele kommerzielle Tools erzeugen zwar Texte in Leichter Sprache, doch ihre Benutzeroberflächen, Hilfetexte und Anleitungen zur Bedienung sind häufig nicht barrierefrei gestaltet. Die eigentliche Zielgruppe kann diese Angebote daher oft nicht nutzen, da Navigation und Inhalte nicht auf ihre Bedürfnisse abgestimmt sind. Ähnliche Probleme zeigen sich in der Forschung: Zwar werden betroffene Personen teilweise einbezogen, doch die entwickelten Lösungen orientieren sich meist an institutionellen Vorgaben – weniger an den tatsächlichen Bedürfnissen der Nutzer.

3 Konzept

3.1 Anforderungsanalyse

Die Anforderungsanalyse bildet die methodische Grundlage für die anschließende Konzeption und Entwicklung des Prototyps. Ziel ist es, alle relevanten funktionalen und nicht-funktionalen Anforderungen systematisch zu erfassen, um eine Anwendung zu schaffen, die zuverlässig, benutzerfreundlich und barrierefrei ist.

Im Rahmen dieser Analyse werden zwei Anforderungskategorien unterschieden:

- **Funktionale Anforderungen** beschreiben die konkreten Aufgaben und Funktionen der Anwendung – also *was* die Anwendung leisten soll.
- **Nicht-funktionale Anforderungen** betreffen qualitative Merkmale – also *wie gut* oder *unter welchen Bedingungen* die Anwendung ihre Aufgaben erfüllen soll.

Die Ableitung der Anforderungen erfolgt in drei Schritten:

1. Eine **visuelle Wettbewerbsanalyse** zeigt, wie Leichte Sprache auf bestehenden Webseiten visuell gestaltet und funktional umgesetzt ist.
2. Eine **Barrierefreiheitsanalyse** auf Grundlage der WCAG 2.2 identifiziert konkrete Anforderungen an Zugänglichkeit, Verständlichkeit und Bedienbarkeit.
3. Die gewonnenen Erkenntnisse werden in strukturierter Form als **User Stories** formuliert, um typische Anforderungen aus Sicht der Nutzer abzubilden.

3.1.1 Visuelle Wettbewerbsanalyse

Die visuelle Wettbewerbsanalyse untersucht, wie Leichte Sprache auf ausgewählten Webseiten gestaltet wird. Dabei werden Merkmale wie Schriftart, Kontrast und Layout anhand festgelegter Kriterien verglichen. Ziel ist es, typische Gestaltungsmuster zu erkennen und daraus Rückschlüsse für die eigene Umsetzung abzuleiten.

Tabelle 3.1: Vergleich von Webseiten in Leichter Sprache

Kriterium	**Netzwerk LS**	**Lebenshilfe**	**Inclusion Europe**
Schriftarten	Fira Sans	Montserrat	gilroy-bold, Open Sans
Serifenlos	ja	ja	ja
Kontrast (Text / Hintergrund, Verhältnis)	Schwarz (#1B1919) / Weiß (#FFFFFF), 17,5:1	Schwarz (#000000) / Weiß (#FFFFFF), 21:1; Blau (#015FA9) / Weiß (#FFFFFF), 5,1:1	Schwarz (#000000) / Weiß (#FFFFFF), 21:1
Schriftgröße (Überschrift / Zwischenüberschrift (ZWÜ) / Fließtext)	32 px/28 px/18 px	40 px/19,2 px/19,2 px	60 px/30 px/23,8 px
Hervorhebungen	Fettung, Unterstreichung, Aufzählungen	Fettung, Blaue Markierung, Aufzählungen	Fettung, Unterstreichung, Gelbe Markierung
Keine Großschreibung	ja	ja	ja
Zeilenabstand	36 px	24 px	42,84 px
Zeilenlänge	kurz	kurz	kurz
Neue Zeile je Satz	ja	ja	ja
Viele Absätze	ja	ja	ja
Textausrichtung	links	links	links
Navigation	groß, eindeutig, kontrastreich	mittelgroß, nicht eindeutig (Titel abgeschnitten), nicht kontrastreich (Pfeil)	mittelgroß, kontrastreich, nicht eindeutig (Ziel fehlt)
Features	Bilder zwischen Text/rechts neben Text	Einleitungsbild, Vorlesefunktion, Schrift vergrößern, Wörterbuch, Übersetzung	Bilder links neben dem Text, Logo, Erklärungsvideos
Positiver Eindruck	Bilder, sehr strukturiert, lesefreundlich, sehr kontrastreich	Bilder, sehr strukturiert, lesefreundlich, kontrastreich, Features	Bilder, sehr strukturiert, sehr lesefreundlich, kontrastreich
Negativer Eindruck	–	Schwacher Kontrast (Bildunterschrift 2,6:1), ZWÜ ohne Strukturelement	Gelbe Markierung, Titel ohne Strukturelement
Quelle	[Net25]	[Leb25]	[Inc25]

Die Analyse zeigte, dass durchgehend eine serifenlose und einheitliche Schriftart wie Open Sans verwendet wurde. Drei unterschiedlich große Schriftgrößen für Überschriften, Zwischenüberschriften und Fließtext sorgten für eine gut erkennbare Textstruktur. Großzügige Zeilenabstände verbesserten die Lesbarkeit. Wichtige Inhalte wurden durch Fettdruck, Unterstreichungen, Farben oder Aufzählungen hervorgehoben. Der Text war linksbündig ausgerichtet, in kurzen Zeilen verfasst und durch Absätze gegliedert. Neue Aussagen begannen jeweils in einer eigenen Zeile. Die Navigation war deutlich, kontrastreich und klar beschriftet. Funktionen wie eine Vorlesehilfe, eine vergrößerbare Schrift und ein integriertes Wörterbuch unterstützten die Zugänglichkeit zusätzlich.

3.1.2 Barrierefreiheitsanalyse

Neben der visuellen Gestaltung stellt die Einhaltung von Barrierefreiheitsstandards ein zentrales Qualitätsmerkmal dar. Die Analyse orientierte sich an den WCAG 2.2 (vgl. Abschnitt 2.2.4). Als praxisnahes Modell diente das BIK-Web-Testverfahren, das die WCAG-Kriterien in überprüfbare Anforderungen überführt [BIK25]. Ziel ist es, daraus konkrete Vorgaben abzuleiten, die eine barrierefreie Nutzung für alle ermöglichen.

Tabelle 3.2: WCAG 2.2 Prüfschritte nach Kategorien (Kurzübersicht)

WCAG-Kategorie	**Details**
9.1.1 Textalternativen	Alt-Texte für interaktive Elemente und Grafiken
9.1.3 Anpassbar	Überschriften h1-h6; Listen strukturiert; Inhalt gegliedert; Formularfelder beschriftet; Fokusreihenfolge logisch; Informationen nicht nur farblich; Bildschirmausrichtung frei
9.1.4 Unterscheidbar	Farbunabhängige Buttons; Textkontrast 4.5:1; Text 200 % vergrößerbar; Responsives Design; Buttons/Symbole 3:1 Kontrast; Textabstände anpassbar; Tooltips bedienbar
9.2.1 Tastaturbedienbar	Alles per Tastatur bedienbar; Fokus nicht blockiert
9.2.4 Navigierbar	Sprunglinks vorhanden; Sinnvolle Seitentitel; Tab-Reihenfolge logisch; Verständliche Linktexte; Alternative Zugangswege; Sinnvolle Überschriften; Fokus sichtbar; Fokus nicht verdeckt
9.2.5 Eingabemodalitäten	Sichtbarer Text = technischer Name; Buttons mind. 24×24 px
9.3.1 Lesbar	Hauptsprache angegeben; Fremdwörter ausgezeichnet
9.3.2 Vorhersehbar	Fokus ändert nichts automatisch; Eingaben ändern nichts automatisch; Navigation konsistent; Begriffe konsistent
9.3.3 Eingabeunterstützung	Fehlermeldungen bei leeren Feldern; Eingabefelder beschriftet; Hilfe bei Fehlern; Validierung vor Absenden
9.4.1 Kompatibel	Buttons/Felder korrekt ausgezeichnet; Statusmeldungen per aria-live

Die hier aufgeführten Anforderungen sind für die geplante Anwendung besonders relevant und betreffen zentrale Aspekte wie Tastaturbedienbarkeit, Inhaltsstruktur und Kontraste. Weitere geprüfte Kriterien sind im Anhang A.6 dargestellt und sollten im Sinne eines ganzheitlich barrierefreien Designs ebenfalls berücksichtigt werden.

3.1.3 User Stories

Die folgenden User Stories beschreiben Anforderungen aus Nutzersicht. Sie wurden aus grundlegenden Überlegungen zur Nutzung, der visuellen Wettbewerbsanalyse 3.1.1 und der Barrierefreiheitsanalyse 3.1.2 abgeleitet und sind in sieben Bereiche gegliedert. Auf eine zusätzliche visuelle Darstellung in Form einer Story Map wurde verzichtet, da die Anforderungen bereits systematisch gegliedert und vollständig dokumentiert sind.

Als Nutzer möchte ich ...

Eingabe:

- ... Text eingeben, um ihn zu vereinfachen.
- ... ursprünglichen Text ein- und ausblenden, um ihn vergleichen zu können.

Ausgabe:

- ... den einfachen Text sofort sehen, um schnell das Ergebnis zu erhalten.
- ... vertraute Wörter verwenden lassen, um den Text gut zu verstehen.
- ... kurze Sätze erhalten, um den Inhalt besser zu erfassen.
- ... dass jeder Satz nur eine Aussage enthält, um Inhalte leichter zu verstehen.
- ... den Text weiter vereinfachen können, um ihn noch besser zu verstehen.
- ... klares Feedback erhalten, um Erfolg oder Fehler zu erkennen.

Navigation:

- ... eine verständliche Navigation, um zu wissen, wo ich bin und was ich tun kann.
- ... eine kurze Anleitung, um die Seite richtig zu nutzen.
- ... zur Startseite zurückkehren, um eine neue Vereinfachung zu starten.

Funktion:

- ... die Website wie eine App bedienen, um sie vertraut und einfach zu nutzen.
- ... Texte speichern, kopieren und verwalten, um sie später zu nutzen.
- ... schwierige Wörter erklärt bekommen, um den Text besser zu verstehen.
- ... einzelne Wörter markieren, um sie gezielt erneut vereinfachen zu können.
- ... Texte vorlesen lassen, um sie anhören zu können.
- ... Schriftgröße und Farbschema anpassen, um besser lesen zu können.

Design:

- ... wichtige Wörter hervorgehoben sehen, um sie schneller zu erkennen.
- ... gut lesbaren, großen Text, um ihn leichter lesen zu können.
- ... dass der Text übersichtlich aufgebaut ist, um mich besser zurechtzufinden.

Barrierefreiheit:

- ... die Seite vollständig mit der Tastatur bedienen, um keine Maus zu brauchen.
- ... zugängliche und verständliche Inhalte, um nichts zu verpassen.
- ... zugängliche Fehlermeldungen und Hinweise erhalten, um informiert zu bleiben.

3.1.4 Auswertung der Anforderungen

Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die zentralen Aufgaben der Anwendung in den Bereichen Texteingabe, Ausgabe, Navigation, Datenverwaltung, unterstützende Werkzeuge und Barrierefreiheit. Bei der **Texteingabe** sollen Nutzer eigene Inhalte direkt eingeben und den Ursprungstext bei Bedarf ein- und ausblenden können. Die **Ausgabe** zeigt die vereinfachte Version sofort nach der Verarbeitung und liefert klare Rückmeldungen. Eine weitere Vereinfachung soll möglich sein. Die **Navigation** soll verständlich und einheitlich sein, kurze Anleitungen bieten und eine Rückkehr zur Startseite ermöglichen. Die **Datenverwaltung** umfasst das Speichern, Anzeigen, Löschen und Kopieren vereinfachter Texte. **Hilfswerkzeuge** unterstützen das Textverständnis, etwa durch Markierungen zur Nachvereinfachung, eine Wörterbuchfunktion und Vorlesen. **Barrierefreiheit** wird durch Tastaturbedienung, zugängliche Inhalte und Rückmeldungen für Screenreader gewährleistet.

Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben, wie die Anwendung ihre Funktionen qualitativ umsetzt – insbesondere in Bezug auf Benutzerfreundlichkeit, Gestaltung, Verständlichkeit und Barrierefreiheit. **Benutzerfreundlichkeit:** Die Navigation soll klar, konsistent und leicht verständlich sein. Eine vertraute, intuitiv bedienbare Oberfläche sowie Einstellmöglichkeiten wie Schriftgrößenanpassung und Kontrastmodi unterstützen eine flexible Nutzung. **Textgestaltung und Lesbarkeit:** Inhalte sollen übersichtlich, strukturiert und gut gegliedert sein. Serifenlose Schrift in ausreichender Größe sowie starker Farbkontrast verbessern die Lesbarkeit. **Sprachliche Qualität:** Die Texte sollen einfache, bekannte Wörter enthalten, damit Leser den Inhalt eigenständig erfassen können. Die Sätze sollen kurz sein und jeweils nur eine Aussage vermitteln, um das Verständnis zu erleichtern. **Barrierefreiheit:** Die Anwendung soll auf allen Geräten sowie bei Vergrößerung und hohem Kontrast zuverlässig und barrierefrei funktionieren. Rückmeldungen und dynamische Inhalte müssen klar, stabil und für alle Nutzergruppen zugänglich sein.

3.2 Entwurf der Benutzeroberfläche

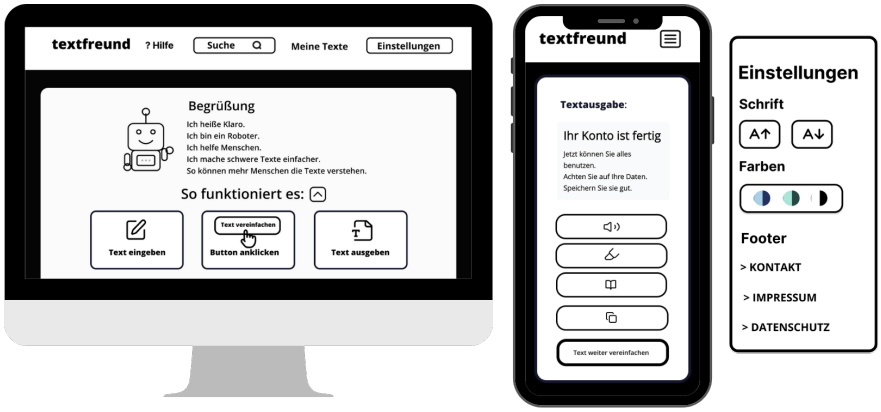
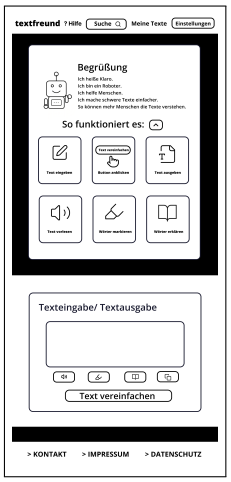


Abbildung 3.1: Darstellung in der Desktop-Ansicht und Mobile-Ansicht.

Der Entwurf der Benutzeroberfläche basierte ausschließlich auf den im vorherigen Kapitel beschriebenen Anforderungen. Das visuelle Design wurde in *Figma*[1] entwickelt und umfasst drei Hauptseiten: die **Startseite** für direkte Textvereinfachung, eine **Archivseite mit Detailseite** zur Verwaltung gespeicherter Texte und eine **Hilfeseite** mit vollständiger Bedienungsanleitung. Alle drei Seiten verfügen über einheitliche Navigationselemente, die eine konsistente Bedienung sowohl im Browser als auch auf mobilen Endgeräten gewährleisten. Zusätzlich wurden exemplarisch Seiten für **Kontakt**, **Impressum** und **Datenschutzerklärung** integriert, die im Anhang A.1 dargestellt sind.

1 Online-Design-Tool, `https://www.figma.com` (Zugriff am 04.06.2025).

1. Startseite



Figma-Vorschau Startseite A.2

Funktion:
Direkte Textvereinfachung

- **Roboter Klaro:** Begrüßung
- **Tutorial:** 6-Kacheln
- **Texteingabe-Feld**
- **Vereinfachen-Buttons**
- **Textausgabe-Bereich**
- **Funktionen:** Wörterbuch, Vorlesen, Markieren, Kopieren

2. Archiv-Seite

Figma-Vorschau Archiv A.3

Funktion:
Verwaltung gespeicherter Texte

- **Vereinfachen-Button**
- **Suchfeld**
- **Listen-Layout:** mit Titel, nach Datum sortiert
- **Auswahl-Funktionen:** Einzeln oder alle markieren/aufheben
- **Lösch-Optionen:** Einzeln oder alle löschen
- **Detailansicht mit Navigation**

3. Hilfe-Seite

Figma-Vorschau Hilfeseite A.2

Funktion:
Vollständige Bedienungsanleitung

- **So ist die Website aufgebaut**
- **So funktioniert die App**

3.2.1 Barrierefreies UI- und UX-Design

Die Gestaltung der Benutzeroberfläche (User Interface, UI) und des Nutzungserlebnisses (User Experience, UX) orientierte sich an den funktionalen und barrierefreien Anforderungen aus den User Stories. Ziel war eine klar strukturierte, gut lesbare und leicht bedienbare Anwendung, die unterschiedliche Bedürfnisse berücksichtigt.

Barrierefreies UI-Design (Benutzeroberfläche)

Die serifenlose Schriftart Open Sans wird in drei deutlich unterscheidbaren Größen verwendet: für Überschriften, Zwischenüberschriften und Fließtext. Alle Texte sind linksbündig ausgerichtet und durch großzügige Zeilenabstände leicht lesbar. Wichtige Begriffe sind fett und farblich hervorgehoben. Das Farbschema erfüllt alle Kontrastanforderungen. Es stehen drei fest definierte Modi zur Auswahl: Schwarz auf Weiß, Grüntöne und Blautöne – jeweils als Kombination aus hellen und dunklen Nuancen. Farben und Schriftgrößen lassen sich über die Barrierefreiheitsleiste im Header anpassen. Buttons und andere Bedienelemente sind klar erkennbar, ausreichend groß, und ihr Fokuszustand ist stets deutlich sichtbar. Die Navigation ist konsistent: Im Header befinden sich Logo, Hilfe, Suche, Archiv und Barrierefreiheitsfunktionen, im Footer die Links zu Kontakt, Impressum und Datenschutz. Auf Mobilgeräten wird der Footer in den oberen Bereich verschoben, um die Zugänglichkeit zu gewährleisten.

Barrierefreies UX-Design (Nutzungserlebnis)

Ein sechsteiliges Tutorial mit der Figur „Klaro“ erklärt den Ablauf der Anwendung in einfacher Sprache. Das Logo ermöglicht jederzeit die Rückkehr zur Startseite. Der zentrale Ablauf entspricht den User Stories: Texteingabe → Vereinfachen → Ausgabe. Der Ausgabebereich erscheint direkt unter dem Eingabefeld und enthält dieselben Werkzeuge: Vorlesen, Wörterbuch, Markieren, Kopieren sowie den Button „Weiter vereinfachen“. Der Ursprungstext kann bei Bedarf ein- und ausgeblendet werden. Archiv erlaubt das Speichern, Suchen und Löschen vereinfachter Texte. Per Klick gelangt man in eine Detailansicht mit bekannter Struktur. Diese erlaubt die erneute Textvereinfachung – analog zur Startseite – und den direkten Wechsel zwischen Einträgen über Navigationspfeile. Die Anwendung ist vollständig per Tastatur bedienbar, mit Sprunglinks und ARIA-Labels für Screenreader. Rückmeldungen erfolgen klar und unmittelbar, z. B. bei Eingabefehlern oder nach erfolgreichem Speichern. Dadurch bleibt die Anwendung in allen Bereichen zugänglich, verständlich und leicht bedienbar.

3.3 Methodisches Vorgehen

3.3.1 Auswahl des Ausgangstextes

Zur Auswahl eines geeigneten Sprachmodells für den Prototyp wird ein Ausgangstext mit typischen sprachlichen Hürden benötigt (vgl. Abschnitt 2.1.2). Nur so lässt sich beurteilen, ob die automatische Vereinfachung im Sinne der Leichten Sprache wirksam ist.

Auswahl und Merkmale

Analysiert wurde der Pressetext *„THM-Studierende fordern: Studium bleibt #unkürzbar“*, der am 18. Juni 2025 auf der Website der Technischen Hochschule Mittelhessen veröffentlicht wurde[1].

THM-Studierende fordern: Studium bleibt #unkürzbar

Erstellt: 18. Juni 2025

Rund 500 Studierende und Mitarbeitende der THM haben sich einer Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen. Unter dem Motto #unkürzbar zogen Demonstrierende vom Campus Eichgärtenallee zum Berliner Platz, wo sie sich mit einem Demo-Zug von Studierenden der Justus-Liebig-Universität zu einer gemeinsamen Abschlusskundgebung vereinten.

Hintergrund sind die Verhandlungen zum Hessischen Hochschulpakt 2026–2031. Die Gespräche zwischen den Hochschulen und dem Land Hessen dauern an – bislang ohne akzeptables Ergebnis. Der Allgemeine Studierendenausschuss (AStA) der THM spricht von drohenden Einschnitten „in Millionenhöhe“ und warnt vor spürbaren Folgen für Beschäftigte und Studierende: eingeschränkte Betreuung, gestrichene Fächer sowie wegfallende Hilfskraftstellen. Auch der GEW Hessen zufolge besteht akuter Handlungsbedarf bei der Hochschulfinanzierung. Ihre Petition „Unterfinanzierung beenden“ wurde bereits von rund 10.000 Unterstützenden unterzeichnet.

Abbildung 3.2: Screenshot des Artikels „THM-Studierende fordern: Studium bleibt #unkürzbar“ (THM, 18.06.2025; vgl. Fußnote).

Der Text erfüllt drei zentrale Kriterien: Er ist (1) **gesellschaftlich relevant**, (2) **öffentlich zugänglich** und (3) **sprachlich komplex**. Letzteres zeigt sich in langen Sätzen, Fachbegriffen und abstrakten Formulierungen.

1 Vgl. THM online: https://web.archive.org/web/20250623144813/https://www.thm.de/site/hochschule/campus/aktuelles/und-ausserdem/thm-studierende-fordern-studium-bleibt-unkuerzbar.html (Zugriff am 23.06.2025)

Analyse und Bewertung

Zur Bewertung kamen vier etablierte Lesbarkeitsformeln zum Einsatz – Flesch-Index, SMOG, LIX und Wiener Sachtextformel –, die den Text übereinstimmend als **schwer verständlich** einstuften [pep25] (vgl. Anhang A.4). Eine anschließende regelbasierte Analyse mit dem GPT-basierten Custom-Modell *Optimeil Prüfer* ergab, dass lediglich 6 von 20 Regeln der Leichten Sprache erfüllt waren [Com25] (vgl. Anhang A.5). Auch die Tools *capito* und *Wortliga* bestätigten diese Einschätzung und identifizierten zahlreiche Verständlichkeitsprobleme [CFS25, WOR25] (vgl. Anhang A.6 und A.7).

> Es zeigt sich, dass der Text zu komplex für Leichte Sprache ist, aber als Ausgangstext für die Modellbewertung genutzt werden kann.

3.3.2 Entscheidungskriterien und Modellauswahl

Die Modellauswahl erfolgt auf zwei Ebenen: strategische Überlegungen zur Architektur (vgl. Abschnitt 2.3.4) und praktische Kriterien zur Umsetzbarkeit. Beide Aspekte fließen in die Auswahl geeigneter Modelle ein.

Strategische Überlegungen

Die automatisierte Textvereinfachung erfolgt in zwei Schritten: Zuerst wird der Ausgangstext vollständig erfasst, anschließend sprachlich vereinfacht, ohne den Sinn zu verändern. Zur Umsetzung dieser Schritte lassen sich zwei Modellarten einsetzen: **Encoder-Decoder-Modelle** und **Decoder-Only-Modelle**. Encoder-Decoder-Modelle lesen den Text vorwärts und rückwärts (bidirektional) und erfassen so den gesamten Zusammenhang. Das ist besonders hilfreich, um auch in komplexen Sätzen den Sinn zu verstehen und korrekt zu vereinfachen. Decoder-Only-Modelle hingegen erstellen den neuen Text Schritt für Schritt von links nach rechts (autoregressiv) und sind besonders gut darin, natürlich klingende Sätze in Leichter Sprache zu formulieren.

Praktische Auswahlkriterien

Im Rahmen dieser nicht-kommerziellen Anwendung stehen nur begrenzte technische Ressourcen zur Verfügung. Deshalb wird ein Modell benötigt, das auf einem **eigenen lokalen Server** betrieben werden kann – ohne Abhängigkeit von kostenpflichtigen Cloud-Diensten oder externen APIs. In Frage kommen nur **vortrainierte Modelle** mit moderatem Rechenbedarf und **offener Lizenz**, die speziell auf die **deutsche Sprache**

ausgelegt sind. Idealerweise unterstützen sie sowohl **Prompting** als auch **Fine-Tuning** und sind über die etablierte Plattformen **Hugging Face** verfügbar. Die Plattform gilt als besonders einsteigerfreundlich, da sie eine umfangreiche Dokumentation, zahlreiche fertige Modelle und praxisnahe Tutorials bereitstellt [Hug23].

Konkrete Modellauswahl

Zur Auswahl geeigneter Sprachmodelle wurde basierend auf den definierten Kriterien eine gezielte Recherche auf Hugging Face durchgeführt. Die Suche fokussierte sich auf **Text2Text-Generation**-Modelle für die **deutsche Sprache** innerhalb der **Transformers**-Bibliothek (vgl. Abbildung A.8). Diese Kombination eignet sich besonders für den Anwendungsfall der Textvereinfachung.

Aus der Recherche gingen **mT5** und **FLAN-T5** als vielversprechende Encoder-Decoder-Modelle hervor. Beide basieren auf der T5-Architektur (Stand: 2023), sind gut dokumentiert, in mehreren Modellgrößen (small bis xxl) verfügbar und gehören zu den am häufigsten heruntergeladenen Modellen in diesen Kategorien.

- **mT5**: multilingual trainiert, darunter Deutsch und so optimiert, dass sprachübergreifende Interferenzen minimiert werden. Dadurch lassen sich deutschsprachige Texte zuverlässig in Leichte Sprache umformulieren [Xue21].
- **FLAN-T5**: basiert auf dem Konzept des instruction-based learning. Es wurde primär auf englischen Texten trainiert, aber zusätzlich per Instruction-Tuning auf verschiedene Aufgabenarten vorbereitet. Dadurch lässt sich das Modell gezielt zur Vereinfachung deutschsprachiger Texte einsetzen [Chu22].

Als Vergleichsmodell wird **LeoLM (Erlesen)** einbezogen – ein deutschsprachiges Decoder-Only-Modell, das gezielt auf Leichte Sprache weitertrainiert wurde [Klö24] (vgl. Abschnitt 2.3.7).

Tabelle 3.3: Ausgewählte Modelle im Vergleich zentraler Kriterien

Kriterium	**LeoLM**	**mT5**	**Flan-T5**
Modellarchitektur	Decoder-only	Encoder-Decoder	Encoder-Decoder
Lokal ausführbar	ja	ja	ja
Open Source	ja	ja	ja
Prompting	ja	eingeschränkt	ja
Fine-Tuning	ja	ja	ja

3.3.3 Experimenteller Aufbau

Testumgebung

Die Tests werden in der cloudbasierten Umgebung **Google Colab Pro**[1] mit **Python** durchgeführt. Diese bietet ausreichende Rechenkapazitäten für die Analyse großer Sprachmodelle. Es werden alle verfügbaren Modellgrößen (Small, Base, Large, XL) der ausgewählten Architekturen getestet. Größere Varianten (z. B. XXL) können aufgrund begrenzter Ressourcen nicht ausgeführt werden. Für die rechenintensiven Modelle kommt eine leistungsstarke **A100-GPU** zum Einsatz.

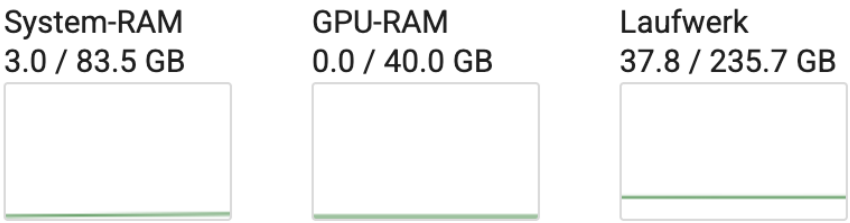


Abbildung 3.3: Verfügbare Hardwareressourcen in Google Colab Pro.

Automatisierter Testablauf

Die automatisierte Testumgebung basiert auf der **Transformers**-Bibliothek von Hugging Face. Sie übernimmt das Laden vortrainierter Sprachmodelle und stellt passende Pipelines zur Verfügung, um Texte zu generieren und zu vereinfachen.

Nach der Installation der notwendigen Bibliotheken

Listing 3.1: Installation benötigter Pakete

```
1 !pip install transformers torch -q
```

werden die Modelle anhand ihrer ID von Hugging Face geladen und gemäß Architekturtyp konfiguriert. Je nach Modellart wird eine entsprechende Pipeline initialisiert:

Listing 3.2: Pipeline für Decoder-Only-Modelle (LeoLM)

```
1 pipe = pipeline("text-generation", model=model_id)
```

Listing 3.3: Pipeline für Encoder-Decoder-Modelle (FLAN-T5, mT5)

```
1 pipe = pipeline("text2text-generation", model=model_id)
```

1 Vgl. online: https://colab.research.google.com/ (Zugriff am 30.06.2025)

Nach Auswahl der passenden Pipeline wird der Prompt zur Textvereinfachung übergeben:

Listing 3.4: Ausführung der Textvereinfachung

```
1 result = pipe(prompt, max_new_tokens=80)
2 output = result[0]["generated_text"]
```

Die Begrenzung auf `max_new_tokens = 80` dient der besseren Vergleichbarkeit zwischen den Modellen und reduziert die Wahrscheinlichkeit halluzinierter Inhalte.

Die folgende Übersicht zeigt den schematischen Aufbau der verwendeten Prompts in den drei Modelltypen.

Vergleich der Prompt-Schemata

FLAN-T5: `Answer the following instruction: {Prompt} Text: {Text} Answer:`

mT5: `{Prompt}: {Text}`

LeoLM: `Aufgabe: {Prompt} Text: {Text} Antwort:`

Der einheitliche Ablauf ermöglicht eine systematische Vergleichbarkeit der Modelle unter identischen Testbedingungen.

Bewertungsmethodik

Die Bewertung der Modellantworten erfolgt in zwei Schritten:

1. **Funktionalitätsprüfung**: Zunächst wird geprüft, ob das Modell überhaupt auf den Prompt reagiert und eine inhaltlich sinnvolle Antwort generiert.
2. **Inhaltsbewertung**: Geeignete Ausgaben werden anschließend im Detail analysiert. Die Bewertung erfolgt sowohl **quantitativ** als auch **qualitativ** (vgl. Abschnitt 2.2.5):
   - **Quantitativ**:
     - Analyse von **Lesbarkeitsmetriken** (z. B. Flesch-Index, Wiener Sachtextformel, LIX, SMOG) mithilfe des *Flesch- Reading-Ease* [pep25]
     - **Regelbasierte Bewertung** mit *Optimeil – LS Prüfer* [Com25]
   - **Qualitativ**:
     - **Manuelle Inhaltsprüfung** durch Vergleich mit dem Ausgangstext hinsichtlich Vollständigkeit und Verständlichkeit der Vereinfachung

3.4 Experimentelle Baseline-Evaluation

3.4.1 Verwendete Prompt-Strategien

Für die Auswertung werden sieben verschiedene Prompts verwendet, die unterschiedliche Strategien zur Textvereinfachung abbilden. Diese lassen sich in drei Kategorien einteilen (vgl. Abschnitt 2.3.4):

Kategorie 1 – Anweisungsbasiertes Lernen:
Prompt 1 (Baseline DE): `Schreibe den Text in leichter deutscher Sprache:`
Prompt 2 (Baseline EN): `Write the text in easy German:`
Prompt 3–4 (Regelbasiert):

```
Vereinfache den Text für Menschen mit Lernschwierigkeiten.
Prompt 3 (Gebot): Verwende einfache Wörter und kurze Sätze.
Prompt 4 (Verbot): Verwende KEINE Fremdwörter, KEINE langen Sätze.
```

Kategorie 2 – Vorlagenbasiertes Lernen:
Prompt 5 (Few-Shot Beispielbasiert):

```
Beispiel 1: Schwer: „Kraftfahrzeug“ → Leicht: „Auto“
Beispiel 2: Schwer: „Der Unterricht findet morgen nicht statt.“
→ Leicht: „Morgen hast du schulfrei.“
Beispiel 3: Schwer: „Die Medikamenteneinnahme erfolgt täglich.“
→ Leicht: „Nimm deine Tabletten jeden Tag ein.“
Vereinfache den Text wie im Beispiel:
Schwer: TEXT Leicht:
```

Kategorie 3 – Stellvertreteraufgaben:
Prompt 6 (Multiple Choice):

```
Wähle den einfachsten Text für Menschen mit Lernschwierigkeiten.
Achte auf einfache Wörter und kurze Sätze.
A) TEXT B) 500 Studenten protestierten gegen Kürzungen.
C) Viele Uni-Leute protestierten.
```

Prompt 7 (Chain-of-Thought):

```
Vereinfache den Text Schritt für Schritt:
Text: TEXT
Schritt 1: Finde schwere Wörter
Schritt 2: Ersetze sie durch einfache Wörter
Schritt 3: Schreibe den vereinfachten Text neu
```

3.4.2 Modellauswertung

Getestet wurden FLAN-T5- (Small–XL), mT5- (Small–XL) und LeoLM-Modelle (7B, 13B) mit den sieben Prompt-Strategien.

Tabelle 3.4: Vergleich der Vereinfachungsleistung der Modelle (vgl. Anhang A.2)

Modell	**Vereinfachungs-ansatz**	**Beste Strategie**	**Typische Probleme**
mT5 (alle)	Keine Vereinfachung	–	Nur Token/Fragmente
FLAN-T5 Small	Sprachmix + Fehler	Baseline Englisch	Deutsch/Englisch Mix, Grammatikfehler
FLAN-T5 Base	Lexikalisch + Fehler	Baseline Englisch	Wortaustausch mit Grammatikfehlern
FLAN-T5 Large	Lexikalisch	Baseline (DE/EN) Regelbasiert	Nur Wörter ersetzt (+ Fehler)
FLAN-T5 XL	Lexikalisch	Baseline (DE/EN) Regelbasiert (nur Gebot)	Nur Wörter ersetzt, etwas besser
LeoLM 7B (Erlesen)	Unzuverlässig	Baseline (DE/EN); Regelb. Verbot	Zahlenfehler, unvollständig
LeoLM 13B (Erlesen)	Echte Vereinfachung	Baseline (DE/EN); Regelbasiert; Few-Shot	Struktur/Wortschatz unzureichend vereinfacht

Funktionalitätsprüfung: Alle **mT5-Modelle** versagten vollständig und produzierten nur unbrauchbare Token wie `<extra_id_0>`. **FLAN-T5**-Modelle zeigten mit zunehmender Größe besseres Sprachverständnis, blieben jedoch auf oberflächliche Wortaustausche beschränkt (z.B. „Studierende"→ „THM-Studenten"). Am besten funktionierten einfache Anweisungen (Baseline-Prompts und zum Teil Regelbasierte-Promts). Nur **LeoLM 13B** erreichte echte strukturelle Vereinfachungen mit Satzaufteilung und Wortschatzanpassung. **LeoLM 7B** erzeugte teilweise brauchbare Ergebnisse, zeigte jedoch Schwächen bei Zahlen und Vollständigkeit. Die besten Resultate erzielten Baseline- und regelbasierte Strategien.

Inhaltsbewertung: Da **mT5**- und **FLAN-T5**-Modelle keine echte Textvereinfachung erreichten, wurde nur Leo 13B vertieft analysiert.

Grundlage war der erste Absatz des Ausgangstextes (vgl. Abschnitt 3.3.1). **LeoLM 13B** erzeugte folgende Vereinfachung, die satzweise verarbeitet wurde:

„Ungefähr 500 Studenten und Mitarbeiter der THM haben bei einer Demo mitgemacht. Sie haben gegen Kürzungen bei den Mitteln für Hochschulen in Hessen demonstriert. Unter dem Motto #unkürzbar gingen Studierende vom Campus Eichgärtenallee zum Berliner Platz. Dort trafen sie sich mit einem Demo-Zug von Studierenden der Justus-Liebig-Universität zu einer gemeinsamen Abschluss-Kundgebung.“

Die **quantitative Bewertung** zeigte eine Verbesserung des Flesch-Werts von 27,2 auf 51,1 (von sehr schwierig zu moderat schwierig) (vgl. Anhang. A.9). Die regelbasierte Bewertung ergab, dass jeder Satz nur eine Information enthält; andere Regeln der Leichten Sprache blieben unverändert umgesetzt (vgl. Anhang. A.10). Die **qualitative Bewertung** zeigte, dass der Inhalt erhalten bleibt, jedoch weiterhin Verständlichkeitshürden bestehen – etwa durch komplexe Begriffe wie „Abschluss-Kundgebungünd lange Satzstrukturen.

> Nur Leo 13B erreicht messbare Verbesserungen und ist für den Prototyp am besten geeignet. FLAN- und mT5-Modelle benötigen gezieltes Nachtraining.

3.4.3 Architektur und Technologiestack

Die mobile Webanwendung zur Textvereinfachung in Leichter Sprache basiert auf einem modularen Technologiestack mit Fokus auf Barrierefreiheit, Wartbarkeit und Skalierbarkeit.

Frontend

- **React mit TypeScript und Vite:** Moderne, typisierte UI-Entwicklung mit schneller Entwicklungsumgebung dank Vite.
- **PWA-Funktionalität:** Installierbar auf Mobilgeräten mit nativer Nutzererfahrung.
- **Tailwind CSS:** Utility-First für responsives, barrierefreies Design.

Backend

- **Node.js mit NestJS:** Modularer Aufbau zur Umsetzung von REST-APIs.
- **SQLite:** Leichtgewichtige Datenbank ohne Server, ideal für den Prototyp.

KI-Service

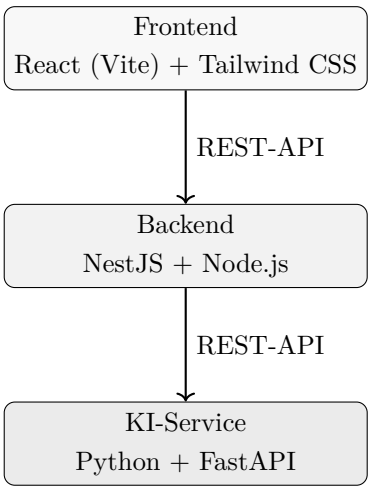
- **Python mit FastAPI (lokal, transformerbasiert):** Eigenständiger Service mit lokal ausgeführtem Sprachmodell über eine REST-Schnittstelle. FastAPI ermöglicht schnelle und moderne API-Entwicklung. Die Trennung vom Hauptsystem erhöht Flexibilität und Wartbarkeit. Externe Cloud-Dienste sind nicht erforderlich.

Zusammenspiel der Komponenten

Komponente	**Aufgabe**	**Technologie**
Frontend	UI, Interaktion, PWA, Barrierefreiheit	React mit Vite (TypeScript), Tailwind CSS
Backend	API, Geschäftslogik, Datenhaltung	Node.js, NestJS (TypeScript), SQLite
KI-Service	Textvereinfachung (lokal)	Python, FastAPI

Systemarchitektur

Die Anwendung folgt einem Drei-Schichten-Modell mit Frontend, Backend und separatem KI-Service. Die Kommunikation erfolgt über REST-APIs. Der KI-Service verarbeitet Texte lokal und kann unabhängig weiterentwickelt werden.



Begründung

Die gewählten Technologien sind etabliert, gut dokumentiert und vereinen Barrierefreiheit, Wartbarkeit und Skalierbarkeit. Die PWA-Umsetzung ermöglicht eine geräteunabhängige Nutzung mit installierbarer App-Oberfläche. Neue KI-Modelle oder Funktionen lassen sich durch die modulare Trennung flexibel integrieren.

4 Fine-Tuning

4.1 Überblick und Zielsetzung

Die Basismodelle FLAN-T5 und mT5 erzielten ohne gezieltes Training keine zufriedenstellenden Ergebnisse bei der Textvereinfachung. Daher wird ein Fine-Tuning durchgeführt, um ihre Leistung zu verbessern und sie mit dem spezialisierten LeoLM-Modell vergleichen zu können. Ziel ist die endgültige Auswahl des besten Modells für den Prototyp.

Das methodische Vorgehen gliedert sich in vier Schritte:

1. **Datengrundlage:** Systematischer Überblick über verfügbare parallele Korpora, Bewertung ihrer Qualität sowie Auswahl und technische Vorbereitung der Trainingsdaten.

2. **Training:** Fine-Tuning der FLAN-T5- und mT5-Modelle auf der vorbereiteten Datengrundlage.

3. **Evaluation:** Quantitative Bewertung anhand von Loss-Werten während des Trainings sowie Vergleich der SARI-Metrik vor und nach dem Fine-Tuning zur Messung der Vereinfachungsqualität.

4. **Modellauswahl:** Vergleichende Analyse der Ergebnisse und endgültige Entscheidung für das leistungsfähigste Modell unter Berücksichtigung quantitativer und qualitativer Kriterien.

4.2 Überblick und Bewertung verfügbarer Datensätze

Für das Fine-Tuning wurden mehrere deutschsprachige Parallelkorpora zur Textvereinfachung analysiert. Die nachfolgende Tabelle zeigt zentrale Unterschiede in Umfang, Herkunft und Alignment-Qualität – also der Genauigkeit der Satzpaar-Zuordnung.

Tabelle 4.1: Übersicht paralleler Korpora für Leichte Sprache [Mad23]

Korpus	Größe	Alignment-Qualität	Herkunft
1. Geasy (2021)	1.087.643 Wörter → 292.552 LS-Wörter	Professionell übersetzt; Qualität: **hoch**	• Verschiedene Textquellen • Professionelle Übersetzer • Standardsprache → Leichte Sprache
2. TextComplexity-DE (2019)	250 Satzpaare	Manuell erstellt; Qualität: **hoch**	• 23 Wikipedia-Artikel • 3 verschiedene Genres • Von Muttersprachlern vereinfacht
3. Toborek et al. (2022)	10.304 Satzpaare [Tob23]	Automatisch erstellt; Qualität: **mittel**	• 7 verschiedene Webseiten • Textextraktion [Tob23] • Nachrichtenartikel • Leichte und Einfache Sprache gemischt
4. APA (2020)	3.616 Satzpaare	Manuell erstellt; Qualität: **hoch**	• Austria Presse Agentur • Einfache Sprache • Nachrichtentexte
5. Klaper et al. (2013)	∼70.000 Tokens	Automatisch erstellt; Qualität: **niedrig**	• 5 öffentliche Webseiten • Textextraktion • Verschiedene Themen

4.2.1 Qualitätsbewertung der Datensätze

Im Detail wurden folgende Korpora untersucht:

- **Geasy-Korpus:** Sehr umfangreich (über 1 Mio. Wörter), professionell übersetzt, hohe Zuordnungsqualität. Enthält ausschließlich Leichte Sprache.
- **TextComplexityDE:** Enthält 250 manuell vereinfachte Satzpaare. Gute Alignment-Qualität, jedoch unklarer Bezug zu den Regeln der Leichten Sprache. Der Datensatz ist auf GitHub[1] verfügbar.
- **Toborek et al.:** Automatisch gesammelter Datensatz mit rund 10.000 Satzpaaren. Inkonsistente Sprache (Leichte/Einfache Sprache) und teils fehlerhafte Zuordnung. Verfügbar auf GitHub[2].
- **APA-Korpus:** Fokus auf Einfache Sprache, keine öffentliche Verfügbarkeit.
- **Klaper et al.:** Sehr geringe Alignment-Qualität, nicht öffentlich verfügbar.

1 https://github.com/babaknaderi/TextComplexityDE (Zugriff: am 25. Juni 2025)
2 https://github.com/mlai-bonn/Simple-German-Corpus (Zugriff: am 25. Juni 2025)

4.2.2 Vorläufige Empfehlung

- **Geasy-Korpus:** Geeignet. Professionell erstellt, regelkonform und mit hoher Zuordnungsqualität. Aufgrund dieser Merkmale stellt der Korpus die bevorzugte Grundlage für die weitere Arbeit dar. Eine offizielle Anfrage zur Nutzung wurde gestellt (vgl. Anhang A.11).
- **Toborek et al.:** Eingeschränkt geeignet. Der Datensatz enthält eine große Menge automatisch extrahierter Satzpaare, jedoch mit fehleranfälliger Zuordnung und gemischten Sprachvarianten (Leichte und Einfache Sprache).
- **TextComplexityDE:** Eingeschränkt geeignet. Der Korpus wurde manuell erstellt, umfasst aber nur 250 Satzpaare. Zudem ist unklar, ob die Vereinfachungen den Regeln der Leichten Sprache entsprechen.
- **Synthetische Daten:** Eingeschränkt geeignet. Die Erstellung manueller Satzpaare ist grundsätzlich möglich, jedoch sehr zeitintensiv und im Rahmen dieser Arbeit nicht zu empfehlen.

4.3 Auswahl der Trainingsdaten

Für das Training werden alle verfügbaren Datensätze aus dem vorherigen Abschnitt verwendet. Der qualitativ hochwertige Geasy-Korpus ist aufgrund fehlender Nutzungserlaubnis nicht verfügbar. Stattdessen kommen mangels Alternativen die verbleibenden Datensätze zum Einsatz: der **Toborek-Korpus**, der **TextComplexityDE**-Datensatz sowie **synthetische Daten**, die mithilfe eines KI-Modells erzeugt werden. Die synthetischen Beispiele erweiterten den Korpusumfang und stellen zusätzliche qualitativ hochwertige Trainingsdaten bereit.

4.3.1 Toborek-Korpus

Der Toborek-Korpus enthält 10.304 parallel ausgerichtete Sätze in Standardsprache sowie in Einfacher oder Leichter Sprache [Tob23].

Das zugehörige Repository wurde automatisiert ausgeführt, um Satzpaare herunterzuladen, zuzuordnen und zu bewerten. Trotz automatisierter Prüfung auf Übereinstimmung und Vollständigkeit konnten lediglich **1.321** qualitativ hochwertige Satzpaare identifiziert werden. Da die Zuordnung nicht ausreichte, wurde der Datensatz anschließend vollständig manuell aufgebaut. Die Verknüpfung von Standardsprache (AS) und Leichter Sprache (LS) erfolgte über identische Dateinamen (z. B. `AS / LS`). Artikel der „Apotheken-Umschau“ und Beiträge der „taz“ wurden ausgeschlossen, da sie entweder nicht in Leichter Sprache vorlagen oder nicht eindeutig zugeordnet werden konnten.

Für die Auswertung wurde ausschließlich die Standardsprache (AS) gezählt. Da ein einzelner Satz in AS häufig in mehrere vereinfachte Sätze in Leichter Sprache (LS) überführt wird, liegt eine **1:n-Zuordnung** vor. Die Tabelle bezieht sich daher nur auf die Ausgangssätze in Standardsprache – nicht auf die Anzahl der LS-Sätze. Insgesamt wurden **4.497 AS-Sätze** aus **262 Artikeln** extrahiert:

Tabelle 4.2: Verteilung der Artikel und AS-Sätze im Toborek-Korpus

Quelle	**Artikel**	**AS-Sätze**
behindertenbeauftragter	20	477
brandeins	46	542
koeln	61	2.185
lebenshilfe	25	412
mdr	95	102
sozialpolitik	15	779
Gesamt	**262**	**4.497**

Die detaillierte technische Umsetzung ist im Anhang dokumentiert (vgl. Anhang A.3).

4.3.2 TextComplexityDE-Datensatz

Der TextComplexityDE-Datensatz wurde zunächst in Betracht gezogen, um rund 250 zusätzliche Satzpaare für das Training zu gewinnen. Seine inhaltliche und strukturelle Qualität gilt als besonders hoch, da die Vereinfachungen nicht nur manuell erstellt, sondern direkt von Betroffenen verfasst wurden [Nad19]. Der Datensatz war frei verfügbar und ließ sich problemlos über GitHub herunterladen. In der praktischen Umsetzung traten jedoch strukturelle Schwierigkeiten auf: Die mitgelieferte CSV-Datei war uneinheitlich formatiert, und es fehlten eindeutige Trennzeichen zur automatisierten Zuordnung von Ausgangs- und Zieltexten. Trotz mehrerer Versuche ließ sich der Datensatz nicht zuverlässig auslesen und wurde daher nicht weiterverwendet.

4.3.3 Synthetische Datensätze

Zur Ergänzung des Korpus wurde ein eigener Datensatz mit synthetischen Satzpaaren erstellt. Als Ausgangsmaterial dienten Texte in Leichter Sprache der Bundeszentrale für politische Bildung (bpb) und des Bundesministeriums für Arbeit und Soziales (BMAS) [Bun14, Bun25b, Bun25a]. Diese Texte wurden mithilfe eines Custom-GPT-Modells automatisiert in schwerere Sprachformen überführt [Kre25] (vgl. Anhang A.13). Dabei wurde auf schwierige Wörter, förmliche Sprache und komplexe Sätze geachtet. Die sogenannte Rückübersetzung stellt sicher, dass die leichten Ausgangstexte authentisch bleiben, während die komplexen Zieltexte künstlich erzeugt werden. Dies gewährleistet eine hohe Qualität des Trainingsmaterials für das Fine-Tuning.

Tabelle 4.3: Verteilung der AS-Sätze im synthetischen Korpus

Quelle	Datei	Artikel	AS-Sätze
BMAS	a640l-mindestlohn	1	191
BMAS	a711l-arbeitsrecht	1	101
bpb	bpb-ueberuns	1	98
Gesamt		**3**	**390**

4.3.4 Gesamtdatenbestand

In den finalen Trainingskorpus wurden insgesamt **265 Artikel** mit **4.887 Sätzen in Standardsprache (AS)** aufgenommen. Sie stammen aus realen Quellen wie dem Toborek-Korpus sowie aus synthetisch erzeugten Texten auf Basis von bpb- und BMAS-Material. Eine Übersicht der Einzelverteilungen ist den vorherigen Abschnitten zu entnehmen.

4.4 Vorbereitung der Trainingsdaten

Für das Fine-Tuning war ein konsistenter und bereinigter Datensatz erforderlich. Dazu wurden die ausgewählten Korpora in mehreren Schritten per Python-Skript aufbereitet (vgl. Anhang A.3).

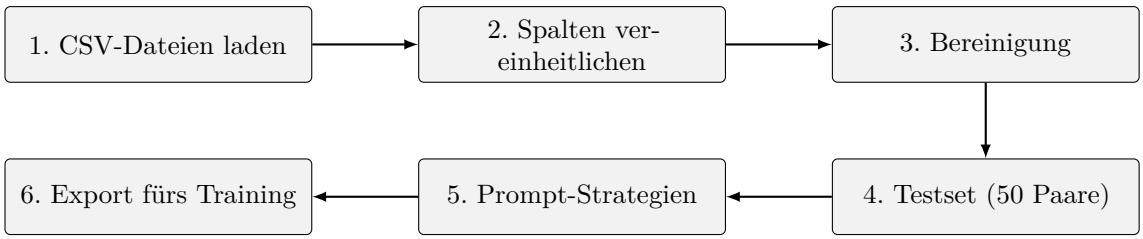


Abbildung 4.1: Ablauf der Datenaufbereitung

Schritt 1–2: Einlesen und Vereinheitlichung. Alle verfügbaren CSV-Dateien wurden geladen und die relevanten Spalten `AS` (Standardsprache) und `LS` (Leichte Sprache) in `input` und `target` überführt.

Schritt 3: Technische Bereinigung. Entfernung von HTML-Tags und mehrfachen Leerzeichen zur Sicherstellung einer sauberen Datenbasis.

Schritt 4: Erstellung eines Testsets. 50 manuell ausgewählte Satzpaare wurden als einheitliches Testset definiert. Zusätzlich wurde eine Spalte `input_prompted` ergänzt, in der die Eingabetexte – je nach Modell – mit einem passenden Prompt versehen

wurden. Je nach Modelltyp unterschieden sich die Prompt-Formate: Während FLAN-T5 ein *instruction-style*-Schema mit Einleitung und Antwortfeld („`Answer:`“) verwendete, folgte mT5 einem *completion-style*-Ansatz, bei dem die vereinfachte Version als direkte Fortsetzung des Eingabetexts formuliert wurde.

Schritt 5: Einsatz von Prompt-Strategien. Diese drei Strategien erzielten die besten Ergebnisse bei den funktionsfähigen Modellen und eignen sich daher besonders für das Fine-Tuning (vgl. Abschnitt 3.4.2).

- **Baseline-Prompt (DE)** – *„Schreibe den Text in leichter deutscher Sprache“* (40 %)
- **Regel-Prompt (Gebot)** – *„Vereinfache den Text für Menschen mit Lernschwierigkeiten. Verwende einfache Wörter und kurze Sätze“* (35 %)
- **Regel-Prompt (Verbot)** – *„Vereinfache den Text für Menschen mit Lernschwierigkeiten. Verwende keine Fremdwörter und keine langen Sätze“* (25 %)

Die Prompts wurden – wie bereits im Testset (vgl. Schritt 4) – modellabhängig formuliert, um eine konsistente Evaluations- und Trainingsbasis sicherzustellen.

4.5 Modelltraining

Für das Fine-Tuning wurden die Large-Varianten von **FLAN-T5** und **mT5** ausgewählt, da sie ein ausgewogenes Verhältnis zwischen Leistungsfähigkeit und Ressourcenbedarf bieten. Auf ein Training der XL-Modelle wurde aufgrund begrenzter GPU-Ressourcen verzichtet.

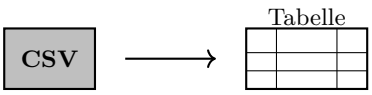
4.5.1 Installation der Bibliotheken

Für das Training wurden die Bibliotheken `transformers`, `datasets`, `accelerate` und `evaluate` installiert. Zur Berechnung der SARI-Metrik waren zusätzlich `sacrebleu` und `sacremoses` erforderlich, da SARI intern BLEU-Komponenten und eine standardisierte Tokenisierung nutzt.

4.5.2 Trainingspipeline

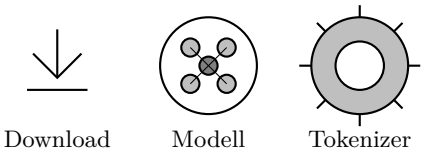
Das Fine-Tuning der Modelle erfolgte in fünf standardisierten Schritten, um eine einheitliche und reproduzierbare Trainingsumgebung sicherzustellen:

1. Datenvorbereitung



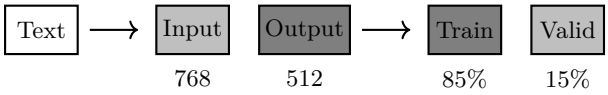
Die Trainingsdaten wurden aus einer CSV-Datei geladen und in das für Hugging Face geeignete Format `datasets.Dataset` überführt. Dieses Format erleichtert die anschließende Tokenisierung und Weiterverarbeitung im Training.

2. Modellinitialisierung



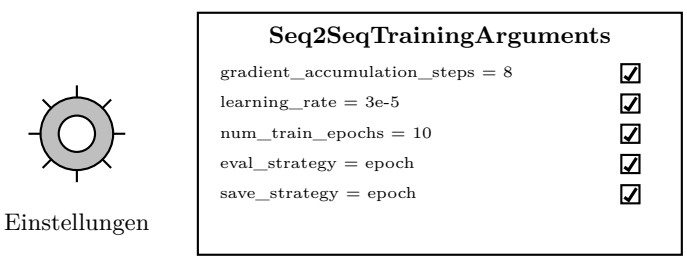
Über die `Transformers`-Bibliothek wurden das jeweilige Modell und der zugehörige Tokenizer geladen. Bei Encoder-Decoder-Architekturen (wie FLAN-T5 und mT5) kam `AutoModelForSeq2SeqLM` zum Einsatz, während für Decoder-Only-Modelle (LeoLM) theoretisch `AutoModelForCausalLM` verwendet werden würde.

3. Tokenisierung und Datenaufteilung



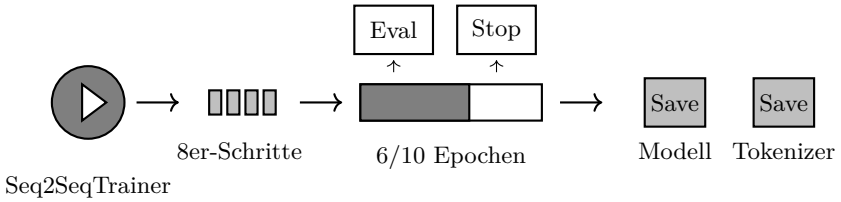
Die Tokenisierung erfolgte über eine benutzerdefinierte Preprocessing-Funktion, bei der Eingabe- und Zieltexte auf eine maximale Länge gebracht wurden (Input: 768 Tokens, Zieltext: 512 Tokens). Diese Werte wurden nach Analyse der längsten Artikel im Datensatz festgelegt, um eine vollständige Abdeckung der Trainingsdaten bei gleichzeitig effizienter Ressourcennutzung zu gewährleisten. Anschließend wurde der Datensatz im Verhältnis 85 % zu 15 % in Trainings- und Validierungsdaten aufgeteilt.

4. Trainingskonfiguration



Die zentralen Trainingsparameter wurden mithilfe von `Seq2SeqTrainingArguments` definiert, um einen stabilen und speicherschonenden Ablauf sicherzustellen. Im Training wurden jeweils acht Beispiele gleichzeitig verarbeitet, um das Modell effizient zu optimieren. Die Lernrate wurde bewusst niedrig gewählt, damit das Modell behutsam an die neuen Daten angepasst wurde. Das Training lief über maximal zehn Durchläufe (Epochen) mit Überprüfung des Verlusts (Loss) und Speicherung nach jeder Epoche.

5. Trainingsdurchführung und Speicherung



Nachdem alle Trainingsparameter festgelegt worden waren, wurde das Training mithilfe des Objekts `Seq2SeqTrainer` gestartet. Dieses übernahm den gesamten Ablauf: das eigentliche Training, die regelmäßige Evaluation auf dem Validierungsdatensatz sowie die Anwendung des `EarlyStoppingCallbacks`, der das Training automatisch beendete, sobald sich die Validierungsleistung über zwei Epochen hinweg nicht verbesserte.

Nach Abschluss des Trainings wurden sowohl das feinjustierte Modell als auch der zugehörige Tokenizer gespeichert, um sie für die spätere Modellanwendung (Inferenz) oder Weiterverwendung bereitzustellen.

4.6 Fine-Tuning-Evaluation

4.6.1 Bewertungsmethode: Die SARI-Metrik

Zur Beurteilung des Fine-Tunings wurde der **SARI-Score** genutzt, der die Qualität der Vereinfachung anhand der Übereinstimmung mit Originaltext, KI-Ausgabe und Referenztexten misst. Er analysiert gezielt drei Aspekte:

- **Beibehaltung:** Welche Wörter wurden sinnvoll übernommen?
- **Löschung:** Welche Wörter wurden korrekt entfernt?
- **Hinzufügung:** Welche Wörter wurden sinnvoll eingefügt?

> Der SARI-Wert reicht von 0 bis 100; höhere Werte zeigen bessere Vereinfachungen.

Die Berechnung des SARI-Scores erfolgte mithilfe der Bibliothek `evaluate` aus dem Hugging-Face-Ökosystem [Xu16] und basierte auf der im vorherigen Kapitel erstellten Testset-Datei (vgl. Abschnitt 4.4; siehe Anhang A.20). Dabei wurde aus der Spalte `input_prompted` jeweils eine Modell-Ausgabe erzeugt, die anschließend mit dem Original- sowie dem Referenztext verglichen wurde.

4.6.2 Ergebnisse vor dem Fine-Tuning

SARI WERT: FLAN-T5-LARGE	**SARI WERT: mT5-LARGE**
SARI-Score: 35.01	SARI-Score: 41.25

Obwohl mT5-Large mit einem SARI-Wert von 41,25 besser abschnitt als FLAN-T5-Large (35,01), stehen die qualitativen Ergebnisse im Widerspruch zu diesen quantitativen Metriken (vgl. Abschnitt 3.4.2). Das mT5-Modell erzeugte unvollständige Ausgaben mit zahlreichen Platzhalter-Token. Die hohe SARI-Bewertung lässt sich vermutlich dadurch erklären, dass das Modell größere Textabschnitte entfernte, was von SARI positiv bewertet wird, obwohl dadurch keine echte Vereinfachung erzielt wurde.

4.6.3 Ergebnisse nach dem Fine-Tuning

Zur Beurteilung des Fine-Tunings und zur Kontrolle möglicher Überanpassung (Overfitting), werden nach jeder Trainingsepoche zwei wichtige Metriken gemessen: Training Loss und Validation Loss.

- **Training Loss:** Gibt an, wie gut das Modell die Trainingsdaten (222 Beispiele) vorhersagt. Ein sinkender Wert zeigt, dass das Modell diese Beispiele zunehmend besser reproduzieren kann.
- **Validation Loss:** Gibt an, wie gut das Modell auf Validierungsdaten (40 Beispiele) reagiert, die es vorher nicht gesehen hat. Ein stabiler oder sinkender Wert zeigt, dass es das Gelernte gut auf neue Daten übertragen kann (Generalisierung).

Trainingsverlauf

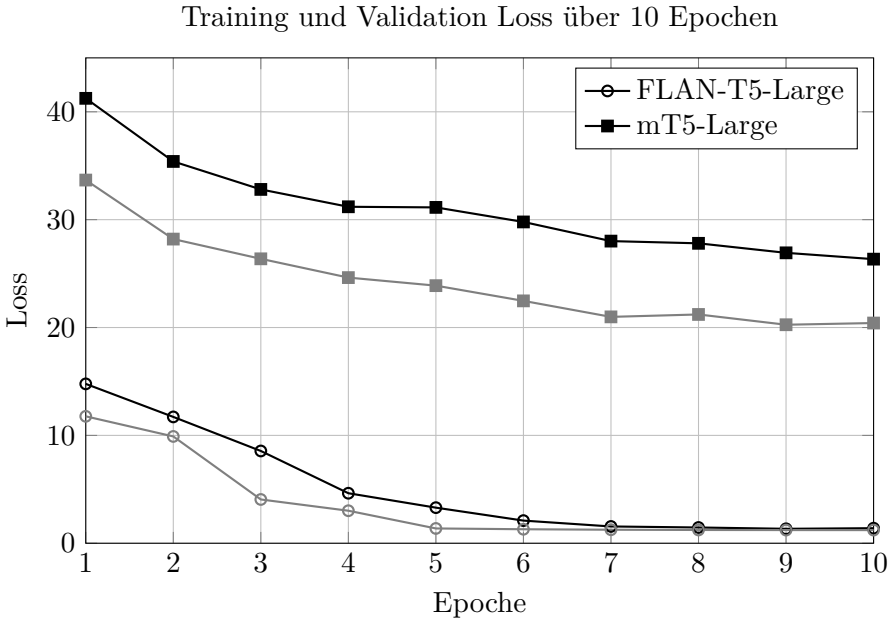


Abbildung 4.2: Verlauf von Trainings- (schwarz) und Validierungsverlust (grau) über 10 Epochen für FLAN-T5 (Kreis) und mT5 (Quadrat).

Beide Modelle zeigen einen stabilen Trainingsverlauf: Sowohl der Training Loss als auch der Validation Loss nehmen über alle zehn Epochen hinweg kontinuierlich ab. Die parallele Entwicklung beider Kurven deutet darauf hin, dass kein Overfitting aufgetreten ist. Während FLAN-T5 bereits nach wenigen Epochen eine starke Reduktion der Fehlerwerte und eine frühe Stabilisierung zeigt, verläuft der Lernprozess bei mT5 etwas langsamer und gleichmäßiger. Insgesamt bestätigen die Ergebnisse, dass beide Modelle zuverlässig lernen und die Trainingsdaten sinnvoll verallgemeinern können.

Sari-Wert

SARI WERT: FLAN-T5-LARGE	**SARI WERT: mT5-LARGE**
SARI-Score: 45.09	SARI-Score: 40.45

Nach dem Fine-Tuning erreicht FLAN-T5-Large mit einem SARI-Wert von 45,09 die höchste Vereinfachungsleistung. Dies entspricht einer Steigerung von etwa 28,7 % gegenüber dem Ausgangswert von 35,01 vor dem Fine-Tuning. mT5-Large zeigt mit 40,45 einen leichten Rückgang von ca. 1,9 % gegenüber dem Ausgangswert.

Qualitative Auswertung

Die qualitative Auswertung nach dem Fine-Tuning zeigt keine wesentlichen Änderungen gegenüber der vorherigen Analyse (vgl. Abschnitt 3.4.2). mT5 bleibt unbrauchbar und erzeugt weiterhin viele fehlerhafte Ausgaben mit Platzhalter-Tokens. FLAN-T5 zeigt leichte Verbesserungen, insbesondere bei der regelbasierten Gebot-Strategie. Dabei wurde der Satz in eine aktive Form umgewandelt und dadurch formal vereinfacht; es treten jedoch grammatikalische Fehler auf, und der Inhalt wird teilweise verändert.

Original:
„Rund 500 Studierende und Mitarbeitende der THM haben sich einer Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.“

Modell-Ausgabe (FLAN-T5 Large, Gebot):
„Die Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen hat rund 500 Studierende und Mitarbeitende der THM angeschlossen.“

4.6.4 Endgültige Modellwahl

FLAN-T5 zeigte durch das Fine-Tuning deutliche Verbesserungen in den quantitativen Metriken (SARI-Score: 35,01 → 45,09), erzeugte jedoch weiterhin keine qualitativ zufriedenstellenden Vereinfachungen im Sinne der Leichten Sprache. Die mT5-Modelle lieferten auch nach dem Fine-Tuning unbrauchbare Ergebnisse mit Platzhalter-Tokens. Obwohl LeoLM in der vorherigen Bewertung (vgl. Abschnitt 3.4.2) die besten Resultate erzielte, erwies sich eine lokale Implementierung als technisch nicht realisierbar. Sowohl LeoLM 7B als auch 13B überforderten die verfügbare Hardware-Infrastruktur. Für die praktische Umsetzung im Prototyp wurde daher **FLAN-T5-Large** verwendet, obwohl es keine echte Textvereinfachung in Leichter Sprache erreicht. Es stellt lediglich einen Proof-of-Concept dar, der die grundsätzliche Machbarkeit einer KI-gestützten Textvereinfachung demonstriert. Eine produktive Anwendung würde leistungsfähigere Hardware oder cloud-basierte Lösungen erfordern.

5 Implementierung

Die vollständige Implementierung der Webanwendung ist als Open-Source-Projekt unter folgender Adresse verfügbar:
`https://git.thm.de/splm63/ba-text-simplification-ai`
Lizenz: MIT License

5.1 Frontend-Implementierung

Das Frontend basiert auf den Designentwürfen und erfüllt zentrale Anforderungen an Benutzerfreundlichkeit und Barrierefreiheit. Es bietet eine einheitliche Nutzerführung, responsives Verhalten sowie anpassbare Farben und Schriftgrößen. Die Anwendung läuft lokal auf `Port 3000`, die finale PWA auf `Port 3002`, und kommuniziert per REST-API mit dem Backend.

5.1.1 Navigation und Layout

Abbildung 5.1: Startseite in der mobilen Ansicht

Layout: Die Benutzeroberfläche ist minimalistisch, responsiv und barrierefrei. Sie bietet hohe Kontraste, klare Strukturen und ist per Tastatur, Maus und Screenreader bedienbar.

Navigation: `Header.tsx` enthält Logo (verlinkt zur Startseite), Links zu “Hilfe” und “Meine Texte”, eine globale `SearchBar.tsx` für die Suche sowie ein Einstellungs-Dropdown. Auf kleinen Bildschirmen erscheint ein Burger-Menü. `Footer.tsx` ergänzt Impressum, Datenschutz und Kontakt und wird mobil im Header eingeblendet. Alle

Navigationsbereiche sind semantisch korrekt mit HTML5-Elementen sowie mit ARIA-Attributen versehen und unterstützen assistive Technologien.

Listing 5.1: Struktur der Navigation für Mobile

```
1 <!-- Mobile Navigation -->
2 <header className="block lg:hidden" role="banner" aria-label="HandyMenü">
3   <nav><ul><li>...</li></ul></nav>
4 </header>
```

In den Einstellungen lassen sich Schriftgröße (60–140 %) und Farbschema (Schwarz-Weiß, Blau, Grün) auswählen. Die Einstellungen werden global über **useState** und **data-theme**-Attribute per CSS-Variablen umgesetzt. Alle Varianten erfüllen mit > 10:1 deutlich die WCAG-Kontrastanforderungen.

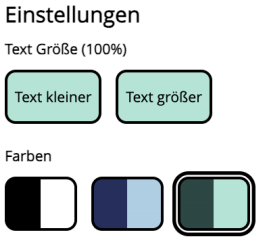


Abbildung 5.2: Mobile Ansicht: Einstellungen für Textgröße und Farbdesign

5.1.2 Progressive Web App (PWA)

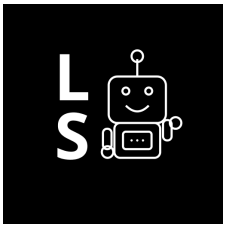


Abbildung 5.3: App-Icon der Anwendung.

Zur plattformübergreifenden Nutzung wurde die Webanwendung als **Progressive Web App (PWA)** umgesetzt. Sie lässt sich auf allen gängigen Geräten (Smartphone, Tablet, Desktop) installieren und verhält sich wie eine native App.

Vite-PWA-Plugin: Die PWA wird über `vite-plugin-pwa` konfiguriert, das automatisch ein `Web-App-Manifest` (Name, Start-URL, Farbschema, Icons) und einen `Service Worker` generiert. Dieser cached statische Ressourcen wie HTML, CSS und JS für die Offline-Nutzung. Funktionen mit Backend-Zugriff (z. B. Textvereinfachung, Archiv, Einstellungen) sind offline nicht verfügbar. Zur plattformspezifischen Darstellung wurden u. a. Apple-Touch-, Android- und SVG-Icons eingebunden. `display: standalone` ermöglicht den Vollbildmodus ohne Adresszeile.

Listing 5.2: Konfiguration des Service Workers via `vite-plugin-pwa`

```
VitePWA({
  registerType: "autoUpdate",
  includeAssets: ["apple-touch-icon.png", "icon.svg"],
  manifest: {
    name: "AI Text Simplification",
    short_name: "TextSimplify",
    start_url: "/",
    theme_color: "#000000",
    background_color: "#000000",
    display: "standalone",
    icons: [...]
    }
})
```

Abbildung 5.4: Lokaler und Netzwerkzugriff der PWA über Port 3002

Backend-Anbindung: Damit die PWA korrekt funktioniert, müssen CORS für Port `3002` aktiviert und das Backend unter `0.0.0.0` erreichbar sein. Alle API-Routen sollten relativ definiert werden, um volle PWA-Kompatibilität zu gewährleisten.

5.1.3 Hauptfunktion: Textvereinfachung

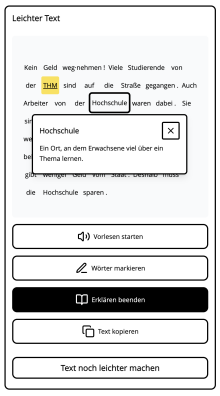


Abbildung 5.5: Mobile Ansicht: Markierungsfunktion und Wörterbuchfunktion

Die zentrale Komponente `TextSimplifier.tsx` steuert den gesamten Vereinfachungsprozess und verwendet die wiederverwendbare `BaseTextComponent.tsx`, die zentrale Anzeige- und Eingabeelemente kapselt. Sie kommt in zwei Varianten zum Einsatz: `TextInput.tsx` für den bearbeitbaren Originaltext und `TextOutput.tsx` für die vereinfachte, nicht-editierbare Ausgabe. Beide enthalten Funktionen wie Vorlesen, Markieren, Wörterbuch und Kopieren.

Kernfunktionen

Backend-Integration:

```
try {
  const response = await saveText(inputHtml, markedWords);
  const simplifiedText =
    response.simplifiedHtml || response.simplifiedText;
  setOutputText(simplifiedText);
  setCurrentTextId(response.id);
  setStatus('Text wurde erfolgreich vereinfacht!', 'success');
} catch (error) {
  setStatus('Fehler beim Vereinfachen des Texts', 'error');
}
```

Das Backend verarbeitet den Originaltext und gibt den vereinfachten Inhalt zurück. Die interaktiven Funktionen im Frontend – wie Vorlesen, Markieren, Wörterbuch und Kopieren – stehen sowohl für den Originaltext als auch für die vereinfachte Version zur Verfügung.

Interaktive Features:

```
// Text vorlesen
window.speechSynthesis.speak(new SpeechSynthesisUtterance(textToRead));

// Schwierige Begriffe markieren (für individuelle Vereinfachung)
const cleanWord = word.toLowerCase().replace(/[.,!?;:]/g, '');
const newMarkedWords = markedWords.includes(cleanWord)
    ? markedWords.filter(w => w !== cleanWord)
    : [...markedWords, cleanWord];

// Definition anzeigen (Wörterbuch)
const cleanWord = word.toLowerCase().replace(/[.,!?;:]/g, '');
const definition = dictionary[cleanWord];
setSelectedWord({ word, definition: definition || "Wort nicht gefunden"
    });

// In Zwischenablage kopieren
navigator.clipboard.writeText(getPlainText(text));
```

Die Markierfunktion ermöglicht es Nutzenden, schwierige Wörter individuell zu kennzeichnen. Diese Auswahl wird an die KI übergeben, um gezielt vereinfacht zu werden. Die Wörterbuchfunktion liefert ergänzende Erklärungen zu diesen Begriffen. Funktionen wie Vorlesen und Kopieren unterstützen zusätzlich das Textverständnis und die Barrierefreiheit.

5.1.4 Benachrichtigungen

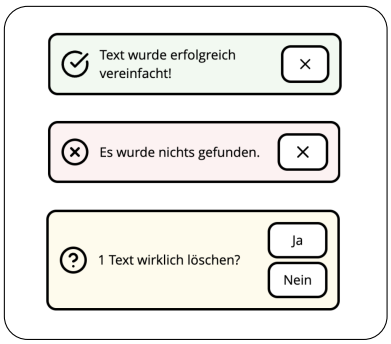


Abbildung 5.6: Erfolgs-, Fehler- und Bestätigungsstatus

Zur nutzerfreundlichen **Rückmeldung** bei Fehlern oder Aktionen wie dem Vereinfachen oder Löschen von Texten wurde ein globales Statussystem auf Basis des *React Context* implementiert. Die visuelle Darstellung erfolgt über eine eigenständige `StatusBanner`-Komponente, die nach einer Nutzeraktion automatisch unterhalb des Headers eingeblendet wird. Die drei Nachrichtentypen **'success'**, **'error'**, **'confirm'** werden visuell klar durch Farbe, Icon und Text unterschieden. Alle Meldungen sind barrierefrei gestaltet, per Tastatur bedienbar und für Screenreader zugänglich (über die Attribute `role="status"` und `aria-live="polite"`). Erfolgs- und Fehlermeldungen verschwinden automatisch nach 30 Sekunden. Das Benachrichtigungsystem unterstützt insbesondere Menschen der Zielgruppe.

Listing 5.3: Kompakte Status-Funktion mit automatischem Entfernen

```
1 const setStatus = useCallback(function(message: string, type: 'success' |
      'error') {
2   const id = `${Date.now()}-${Math.random()}`;
3   const newNotification = { id, message, type, timestamp: Date.now() };
4
5   setNotifications(prev => [...prev, newNotification]);
6
7   // Erfolg und Fehler verschwinden nach 30s automatisch
8   setTimeout(() => removeNotification(id), 30000);
9 }, [removeNotification]);
```

Listing 5.4: Beispielhafte Verwendung von `setStatus`

```
1 setStatus('Fehler beim Löschen des Texts', 'error');
```

5.1.5 Weitere Seiten

Textverwaltung: `SavedTextsPage.tsx` zeigt eine durchsuchbare Übersicht aller gespeicherten Texte mit Mehrfachauswahl zum Löschen; `TextDetailPage.tsx` dient der Bearbeitung einzelner Einträge (vgl. Anhang A.21, A.22).

Hilfestellungen zur Nutzung: Die Figur „Klaro“ auf `HomePage.tsx` führt in die Textvereinfachung ein; `HelpPage.tsx` erklärt in Leichter Sprache die Navigation, zentrale Funktionen und den Ablauf der Textvereinfachung (vgl. Anhang A.23, A.24).

Rechtlich erforderliche Seiten: Die Seiten Kontakt (`ContactPage.tsx`), Impressum (`LegalNoticePage.tsx`) und Datenschutz (`PrivacyPolicyPage.tsx`) wurden exemplarisch umgesetzt, um die Anwendung vollständig rechtlich darzustellen (vgl. Anhang A.25, A.26, A.27).

5.2 Backend-Implementierung

Die Backend-Architektur basiert auf dem TypeScript-Framework NestJS und stellt eine modulare REST-API für KI-gestützte Textvereinfachung mit persistenter Datenspeicherung bereit. NestJS nutzt Dependency Injection und das Decorator-Pattern für strukturierte, testbare API-Entwicklung. Die Architektur trennt klar zwischen Frontend-Integration, Datenbankoperationen und externer KI-Anbindung.

5.2.1 Server-Konfiguration und API-Setup

Die Grundkonfiguration auf `Port 3001` umfasst drei zentrale Komponenten: **CORS** für die Frontend-Kommunikation (im Prototyp mit `origin: true` offen freigegeben), **Cookie-Parser** für sessionbasiertes Benutzer-Management ohne klassische Authentifizierung sowie **Swagger** zur automatischen API-Dokumentation.

Listing 5.5: NestJS-Grundkonfiguration mit CORS und Swagger

```
1 async function bootstrap() {
2   const app = await NestFactory.create(AppModule);
3   app.enableCors({ origin: true, credentials: true });
4   app.use(cookieParser());
5
6   SwaggerModule.setup('api', app, new DocumentBuilder().build());
7   app.listen(3001, '0.0.0.0');
8 }
```

5.2.2 Datenbankanbindung und Datenmodellierung

Für die Speicherung der Daten kommt **SQLite** zum Einsatz – eine leichtgewichtige, lokale Datenbank, die ohne zusätzliche Installation auskommt. Die Anbindung erfolgt über **TypeORM**, das eine strukturierte und objektorientierte Modellierung der Daten ermöglicht. Diese Kombination eignet sich besonders für die Entwicklung von Prototypen, ist jedoch für den produktiven Einsatz nicht ausgelegt. Die Einbindung der Datenbank erfolgt in der Datei `app.module.ts`:

Listing 5.6: TypeORM-Integration im AppModule

```
imports: [
  TypeOrmModule.forRoot({
    type: 'sqlite',
    database: 'db/textfreund.sqlite',
    entities: [UserText, UserPreference],
    synchronize: true,
  }),
  DatabaseApiModule,
  FrontendApiModule,
],
```

Das Datenbankschema umfasst zwei zentrale Entitäten:

Zum einen `user-text.entity.ts`, die die Textdaten eines Benutzers speichert – inklusive Originalversion, vereinfachter Version und einer Liste schwieriger Wörter. Zum anderen `user-preference.entity.ts`, die benutzerspezifische Einstellungen wie Farbschema und Schriftgröße verwaltet.

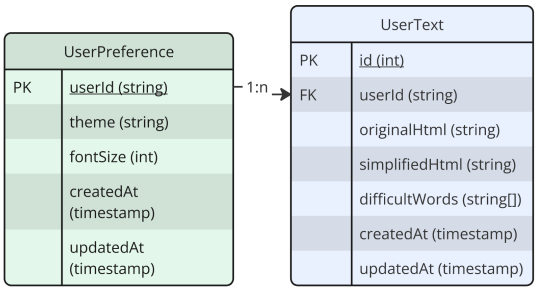


Abbildung 5.7: ERD der Datenbank mit `UserPreference` und `UserText`

5.2.3 Frontend-Backend-Kommunikation über die REST-API

Die Kommunikation zwischen Frontend und Backend erfolgt über eine **REST-API** mit Cookie-basiertem **Session-Management**. Beim ersten Aufruf des `/session`-Endpunkts wird eine eindeutige `userId` generiert, als persistenter Cookie gespeichert und der **Seed-Service** ausgeführt. Dieser legt **Default-Einstellungen** (Farbschema `default`, Schriftgröße `100`) sowie drei Beispieltexte für Testzwecke an.

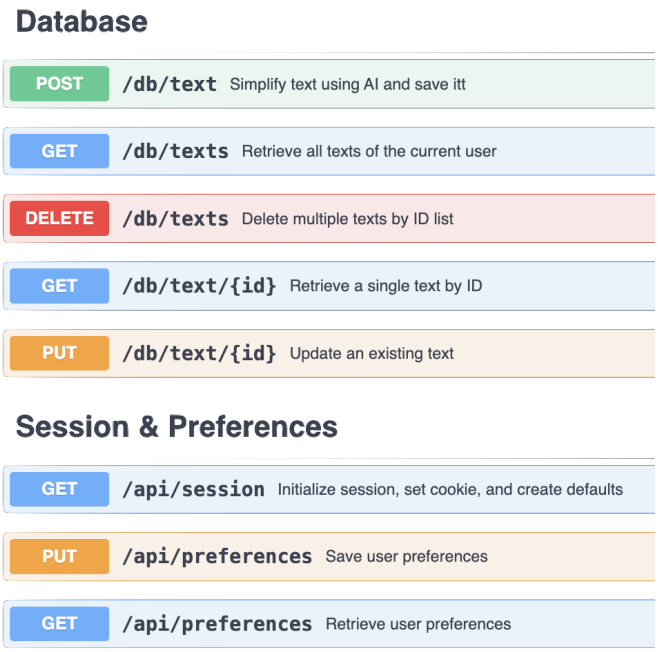


Abbildung 5.8: Alle REST-Endpunkte für Session, Einstellungen und Text-CRUD: Swagger-UI unter `http://localhost:3001/api`

5.2.4 Datenvalidierung

Alle Eingaben werden über **Data Transfer Objects (DTOs)** mit `class-validator` validiert. Die DTOs definieren dabei die erlaubte Struktur und Wertebereiche für den Datenaustausch zwischen Frontend und Backend.

Zum Einsatz kommen folgende DTOs:

- `CreateUserTextDto` – prüft, ob ein Originaltext als `string` übergeben wird (z. B. HTML) sowie optional eine Liste schwieriger Wörter als `string[]`.
- `UpdateUserTextDto` – erlaubt das Aktualisieren einzelner Felder eines vorhandenen Texts; validiert optional übergebene Felder: Originaltext (`string`), vereinfachter Text (`string`) und Wortliste (`string[]`).
- `UpdateUserPreferenceDto` – erlaubt ausschließlich die Änderung von Farbschema (nur `default`, `blue`, `green`) und Schriftgröße (60–140 %).
- `DeleteMultipleDto` – ermöglicht das gleichzeitige Löschen mehrerer Texte über eine Liste von IDs (`number[]`).

5.2.5 Qualitätssicherung durch End-to-End-Tests

Alle zentralen Backend-Funktionen werden durch **End-to-End-Tests** mit Jest (Test-Runner und Assertion-Framework) und Supertest (HTTP-Client zur API-Simulation) abgesichert. Die Tests decken realistische Nutzungsszenarien ab, stärken die Stabilität der Anwendung und helfen, Fehler frühzeitig zu erkennen. Insgesamt **acht Tests** prüfen die komplette REST-API – von der Sitzungsinitialisierung über das Speichern und Abrufen von Texteingaben bis hin zur Bulk-Löschung. Dabei wird über die POST-Route auch die Anbindung des KI-Dienstes zur Textvereinfachung mitgetestet.

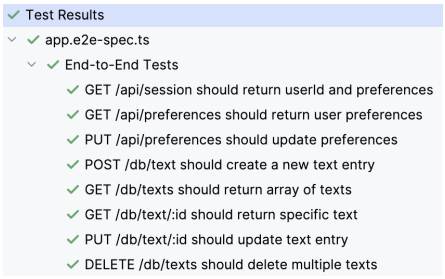


Abbildung 5.9: Erfolgreiche End-to-End-Tests (8/8) in WebStorm

5.3 KI-Implementierung

Die Textvereinfachung erfolgt in einer lokal ausgeführten Python-Umgebung mit dem feinjustierten Modell **FLAN-T5-Large**. Der eigenständige **KI-Service** läuft auf `Port 8000` und verarbeitet HTML- und Plaintext-Inhalte satzweise unter Erhalt der semantischen Struktur. Die Kommunikation mit dem NestJS-Backend erfolgt über einen klar definierten **HTTP-Endpunkt** und trennt Vorverarbeitung, Inferenz und Systemintegration sauber voneinander.

Listing 5.7: Verzeichnis der KI-Komponente im Backend

```
backend/
  .venv/     # Virtuelle Umgebung
  src/
    model/   # Enthält die gesamte KI-Komponente
```

5.3.1 Service-Architektur und API-Anbindung

Der **KI-Service** basiert auf dem Web-Framework `FastAPI`, das eine moderne, asynchrone Architektur mit automatischer Validierung und Dokumentation über `Pydantic` bietet. Die Anwendung wird über `app.py` gestartet. Dort wird die Route `/simplify` mithilfe von `include_router` eingebunden:

Listing 5.8: Einbindung der Route in `app.py`

```
1 app = FastAPI()
2 app.include_router(simplify_router)
```

Die Route `/simplify` nimmt `POST`-Anfragen mit einem Texteingang und optionalen schwierigen Wörtern entgegen. Nach Validierung über `Pydantic` erfolgt die Übergabe an `structured_simplifier.py` zur weiteren Verarbeitung.

Listing 5.9: FastAPI-Route zur Textvereinfachung

```
1 @router.post("/simplify")
2 async def simplify(req: SimplifyRequest):
3     result = simplify_html_text(req.text, req.difficultWords)
4     return {"simplifiedHtml": result}
```

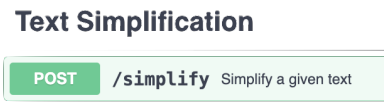


Abbildung 5.10: Automatisch generierte API-Dokumentation für die KI-gestützte Textvereinfachung: Swagger-UI unter `http://localhost:8000/docs`

5.3.2 Integration des Sprachmodells

Das KI-Modul nutzt das lokal gespeichertes Modell FLAN-T5-Large, das über die `HuggingFace-Transformers-Bibliothek` eingebunden wird. Für die Vereinfachung werden zwei Prompt-Strategien unterschieden (vgl. Abschnitt 4.4):

Baseline-Prompt (Deutsch): Die Standardvereinfachung nutzt einen Prompt, der dem Vorgehen entspricht, mit dem das Modell zu 40 % trainiert wurde.

Listing 5.10: Geplanter Prompt zur Standard-Vereinfachung

```
1 prompt = (
2   f"Answer the following instruction: Schreibe den folgenden Text in
        leichter deutscher Sprache.\n"
3   f"Text: {text.strip()}\nAnswer:")
```

Regel-Prompt (Verbot) Bei schwierigen Wörtern wird ein Prompt eingesetzt, der deren Verwendung vermeidet – eine Strategie aus 25 % der Trainingsdaten.

Listing 5.11: Geplanter Prompt zur gezielten Vermeidung schwieriger Wörter

```
1 prompt = (
2   f"Answer the following instruction: Vereinfache den folgenden Text für
        Menschen mit Lernschwierigkeiten."
3   f"Verwende dabei nicht die folgenden schwierigen Wörter: {words_list}."
4   f"Text: {text.strip()}\nAnswer:")
```

> Trotz deutsch formulierter Prompts reagierte das Modell auf englische Instruktionen zuverlässiger. Daher wurden im Prototyp englische Prompts verwendet.

5.3.3 Ablauf der Textvereinfachung

Die Verarbeitungspipeline des KI-Moduls lässt sich in drei abgegrenzte Schritte gliedern – von der Anfrage bis zur strukturerhaltenden HTML-Ausgabe:

1. **API-Aufruf:** Der `POST`-Request an `/simplify` enthält den Eingabetext (HTML oder Plaintext) sowie optional schwierige Wörter.
2. **Validierung und Weiterleitung:** Die Eingabe wird durch `Pydantic` geprüft und an den Vereinfachungsservice übergeben.
3. **Ausgabe:** Die HTML-Ausgabe wird an das Backend zurückgesendet.

Die konkrete Umsetzung dieser Schritte erfolgt im zuständigen KI-Service, der nachfolgend detailliert beschrieben wird.

5.3.4 KI-Service zur HTML-Textvereinfachung

Die zentrale Logik zur Textvereinfachung befindet sich in `structured_simplifier.py` und wird über die Route `/simplify` angesteuert. Dort werden HTML- und Plaintext-Eingaben strukturell analysiert, vereinfacht und als HTML zurückgegeben.

1. HTML-Bereinigung: Mit `BeautifulSoup` wird der Eingabetext geparst und um irrelevante Elemente wie `<script>`, `<style>` oder `<nav>` bereinigt:

Listing 5.12: Entfernung irrelevanter HTML-Tags

```
soup = BeautifulSoup(html_text, "lxml")
for tag in body(["script", "style", "nav", "header", "footer", "aside"]):
    tag.decompose()
```

2. Strukturvereinheitlichung: Die semantische Gliederung bleibt erhalten, wird aber reduziert: nur `<h1>`, `<h2>`, `<p>`, `<ul>` etc. bleiben bestehen; tiefere Überschriften werden in `<p><strong>` umgewandelt.

3. Satzweise Vereinfachung: Mit `spaCy` erfolgt die Satzsegmentierung, anschließend die Vereinfachung jedes Satzes:

Listing 5.13: Satzweise Vereinfachung

```
doc = nlp(text)
for sent in doc.sents:
    simplified = simplify_text(sentence_text, difficult_words)
```

4. Strukturwahrung: Nur definierte HTML-Tags werden rekursiv erhalten:

Listing 5.14: Erlaubte HTML-Tags

```
KEEP_TAGS = {'h1', 'h2', 'p', 'ul', 'ol', 'li', 'strong', 'div'}
```

Hinweis: Liegt Plaintext (kein HTML) vor, wird jeder Satz als <p>-Absatz formatiert:

Listing 5.15: Plaintext zu HTML

```
if not any(tag in html_text for tag in ['<', '>']):
    sentences = [sent.text for sent in nlp(html_text).sents]
    html_text = ''.join(f"<p>{s}</p>" for s in sentences)
```

5.3.5 Modultests für die KI-Vereinfachung

Die gesamte Textvereinfachung – von der API-Anfrage bis zur HTML-Ausgabe – wird durch **zehn** gezielte **Unit Tests** mit `pytest` (Framework für automatisierte Tests in Python) und dem `FastAPI TestClient` (Werkzeug zum Simulieren von API-Aufrufen) abgesichert. Getestet werden sowohl die Funktionalität der API-Endpunkte als auch die interne Verarbeitung: Dazu zählen die Auswahl der passenden Prompt-Strategie bei schwierigen Wörtern, die Satzsegmentierung sowie die strukturierte Umwandlung und Vereinfachung von HTML-Inhalten. Auch komplexe oder fehlerhafte Eingaben (z. B. ungültiges HTML) werden berücksichtigt, um die Robustheit des Systems sicherzustellen.

Abbildung 5.11: Alle Modultests der KI-Vereinfachung erfolgreich durchlaufen

6 Zusammenfassung

6.1 Diskussion

Diese Arbeit zeigt, dass KI-gestützte Textvereinfachung in Leichter Sprache prinzipiell möglich ist – zugleich aber mit erheblichen Herausforderungen verbunden bleibt. Im Folgenden werden zentrale Aspekte der Arbeit kritisch reflektiert: von der problematischen Datenlage über technische und methodische Grenzen bis hin zu Design- und Usability-Herausforderungen.

6.1.1 Datenlage als zentrales Problem

Die Leistungsfähigkeit eines KI-Systems hängt maßgeblich von Qualität und Umfang der Trainingsdaten ab. Für Leichte Sprache existieren jedoch nur wenige frei zugängliche Datensätze, die groß, vielfältig und qualitativ hochwertig sind. Viele stammen aus Behördenkontexten und decken nur ein enges Themenspektrum ab.

Diese Arbeit verdeutlicht das Problem exemplarisch: Der hochwertige Geasy-Korpus war trotz Anfrage nicht nutzbar. Im Toborek-Korpus erwiesen sich von über 10.000 Satzpaaren nur rund 1.300 als automatisch verwertbar. Erst durch aufwendige manuelle Nachbearbeitung konnte etwa die Hälfte davon sinnvoll eingesetzt werden. Weitere Korpora wie TextComplexityDE waren technisch kaum nutzbar. Ergänzend kamen synthetische Datensätze zum Einsatz – qualitativ vielversprechend, aber thematisch eingeschränkt und zeitaufwendig in der Erstellung. Uneinheitliche Standards, Vermischungen mit Einfacher Sprache und rechtliche Unsicherheiten erschwerten die Nutzung zusätzlich.

Ein Vergleich mit dem Projekt *ErLeSen*, das über 8.130 Dokumentpaare umfasste – mehr als 30-mal so viele wie in dieser Arbeit – zeigt den Einfluss großer Datenmengen: Besonders bei LeoLM-13B wurden dort erste klare Ansätze echter Vereinfachung sichtbar.

6.1.2 Technische Grenzen

Eine reine Steuerung über Prompting erwies sich als unzureichend: Kleine Modelle verstanden Deutsch nur eingeschränkt, große Modelle reagierten kaum zuverlässig auf die gestellten Aufgaben. Deshalb rückte das Fine-Tuning in den Mittelpunkt dieser Arbeit – auch, um eine fundierte Vergleichbarkeit mit den LeoLM-Modellen zu ermöglichen.

Doch auch hier stieß die Umsetzung an technische Grenzen: Das Training größerer XL-Modelle war selbst mit starker GPU-Leistung über Google Colab Pro nicht möglich. **mT5-Large** zeigte trotz Fine-Tuning keine spürbaren Verbesserungen. Die Hoffnung lag daher auf **FLAN-T5-Large**, um zumindest erste Fortschritte zu erzielen und eine Grundlage für weiterführende Forschung zur KI-gestützten Vereinfachung zu schaffen. Dabei wäre es auch interessant gewesen, systematisch zu untersuchen, inwieweit die Modellarchitektur Einfluss auf die Vereinfachungsleistung hat – insbesondere, ob Encoder-Decoder-Modelle geeigneter sind als reine Decoder-Only-Modelle, die bislang im Fokus kommerzieller Tools und der Forschung stehen.

Trotz aller Bemühungen blieb die Modellqualität begrenzt – vor allem aufgrund knapper Rechenressourcen und geringer Datenbasis. Die Experimente verursachten rund 400 Recheneinheiten und führten schnell zu Nutzungslimits in Colab Pro. Eine lokale Lösung war ebenfalls nicht umsetzbar: Selbst ein Betrieb von LeoLM-7B als Kompromissmodell im Prototyp scheiterte am iMac M1 mit 16 GB RAM – die CPU-Leistung war unzureichend. Das final implementierte **FLAN-T5-Large** stellt letztlich nur einen **Proof-of-Concept** dar, da es unter den gegebenen technischen Rahmenbedingungen als einziges Modell praktisch einsetzbar war, und bleibt weit hinter den Anforderungen der Leichten Sprache zurück.

Die Ergebnisse verdeutlichen: Für eine verlässliche, KI-gestützte Vereinfachung braucht es nicht nur spezialisierte Modelle und hochwertige Daten, sondern auch deutlich mehr Rechenleistung, als im Rahmen dieser Arbeit zur Verfügung stand.

6.1.3 Methodische Herausforderungen

Die Bewertung der Modellleistung erwies sich als schwierig. SARI-Werte standen teils im Widerspruch zu den qualitativen Ergebnissen: So erzielte **mT5** zwar hohe Werte (41,25), generierte jedoch nur unbrauchbare Token wie `<extra_id_0>`. Auch die Loss-Werte beim Fine-Tuning waren trügerisch – obwohl Trainings- und Validierungs-Loss parallel sanken und kein Overfitting sichtbar war, führte das nicht automatisch zu besseren Vereinfachungen. Hier spiegeln sich auch die Limitationen der verwendeten Datensätze wider.

Für eine vergleichbare Bewertung war die Verwendung einheitlicher Prompt-Formate notwendig – insbesondere bei **mT5** führte dies jedoch zu suboptimalen Ergebnissen. Dies zeigt, dass modellspezifische Anpassungen für valide Ergebnisse entscheidend sind. Hinzu kamen technische Einschränkungen: Die Modelle funktionierten nur bei kurzen Eingaben zuverlässig und brachen bei längeren Texten häufig ab. Als Workaround wurde eine satzweise Verarbeitung mit anschließender rekursiver Zusammenführung implementiert, um diese Längenbeschränkungen zu umgehen.

Die Evaluation basierte zudem auf einem einzelnen Beispieltext aus dem universitären Kontext, was die Übertragbarkeit auf andere Domänen deutlich einschränkt. Eine systematische Bewertung unterschiedlicher Texttypen und -längen konnte aus Zeit- und Ressourcengründen nicht erfolgen.

6.1.4 Design- und Usability-Herausforderungen

Die Anwendung sollte einerseits ein minimalistisches, klar strukturiertes Design bieten, andererseits barrierefrei und funktional genug für die Zielgruppe sein. Die Schnittmenge zwischen einfacher Bedienung und unterstützenden Funktionen – wie Wörterbuch, Markierung oder PWA – erwies sich als problematisch. Exemplarisch zeigt sich das Dilemma bei der PWA-Funktionalität: Ist App-Installierbarkeit hilfreich oder unnötig komplex? Gleichzeitig durfte die Anwendung nicht zu stark vereinfacht werden, um Nutzer nicht auf weniger barrierearmen Webseiten zu überfordern. Es galt, ein Gleichgewicht zwischen Barrierefreiheit und gestalterischer Verantwortung zu finden.

Diese konzeptionellen Herausforderungen spiegelten sich auch in der technischen Umsetzung wider: Kontraste, Tastaturbedienung, Screenreader-Kompatibilität sowie responsive Anpassungen (Farben, Schriftgröße) erforderten hohe Sorgfalt. Zusätzlich musste die gesamte Anwendung in Leichter Sprache gestaltet werden, wofür auf KI-gestützte Übersetzung zurückgegriffen wurde. Ob die Zielgruppe diese Texte tatsächlich versteht, bleibt ebenfalls offen.

6.2 Fazit

Diese Arbeit untersuchte, wie KI-gestützte Textvereinfachung in einer barrierefreien mobilen Anwendung umgesetzt werden kann. Die in Abschnitt 1.2 formulierten **Forschungsfragen** lassen sich wie folgt beantworten:

F1: Eine barrierefreie mobile KI-gestützte Anwendung ist grundsätzlich realisierbar, wie der entwickelte Prototyp zeigt. Die PWA-Lösung mit lokaler KI-Integration, barrierefreier UI/UX und Leichte-Sprache-Features demonstriert die technische Machbarkeit. Jedoch bleiben die Qualitäts- und Ressourcenlimitationen der KI-Textvereinfachung ein entscheidender Faktor – das System erreicht nur Proof-of-Concept-Status.

F1.1: Die Überführung der Leichte-Sprache-Regeln in automatisierte Prozesse ist nur teilweise erfolgreich. Sowohl Prompting als auch Fine-Tuning zeigen deutliche Grenzen auf: Nur oberflächliche Regeln werden umgesetzt. Es bleibt herausfordernd, inhaltliche Aussagen sinnvoll zu vereinfachen, ohne wesentliche Informationen zu verlieren. Ebenso erweist sich die klare strukturelle und visuelle Aufbereitung der Texte als anspruchsvoll.

F1.2: Als zentrale Anforderungen erwiesen sich WCAG 2.2-Konformität, PWA-Funktionalität, barrierefreie Anpassbarkeit und durchgängige Leichte Sprache in der UI/UX. Die Balance zwischen Einfachheit und Funktionalität sowie zwischen Barrierefreiheit und Alltagstauglichkeit bleibt eine Designherausforderung.

Diese Arbeit bietet sowohl praxisnahe Impulse als auch wissenschaftliche Erkenntnisse zur KI-gestützten Barrierefreiheit. Praktisch wurde der erste **Open-Source-Prototyp** zur lokalen KI-gestützten Textvereinfachung entwickelt, der **WCAG 2.2-konforme Barrierefreiheit** mit **responsivem Design** und **PWA-Integration** kombiniert.

Technisch innovativ ist die strukturerhaltende Verarbeitung von HTML und Plaintext mit rekursiver Wiederherstellung der Dokumentstruktur sowie der Workaround der satzweisen Verarbeitung zur Maximierung der KI-Leistung.

Wissenschaftlich zeigt die Arbeit systematisch den Vergleich zwischen Encoder-Decoder- und Decoder-Only-Modellen für deutsche Leichte Sprache auf, deckt die Grenzen des reinen Promptings auf und demonstriert den Widerspruch zwischen SARI-Metriken und qualitativer Bewertung. Besonders relevant ist die Aufdeckung der problematischen Datenlage als systemischen **Flaschenhals**. Die entwickelte Systemarchitektur und gewonnenen Erkenntnisse können als Grundlage für weiterführende Forschung dienen.

Trotz der erreichten Fortschritte bleibt das zentrale Problem bestehen: Ausgerechnet jene Zielgruppen, die am meisten von barrierefreier Kommunikation profitieren würden, können die Anwendung aufgrund der KI-Qualitätsmängel nicht praktisch nutzen. Die Integration eines zuverlässigen Modells scheiterte an begrenzten Hardwareressourcen, begrenzter Datenlage und der daraus resultierenden unzureichenden Modellqualität.

6.3 Ausblick

Zukünftige Entwicklungen sollten betroffene Menschen systematisch einbeziehen. Nur durch Tests mit der Zielgruppe, Screenreader-Nutzern und Barrierefreiheits-Experten lässt sich bewerten, ob KI-Vereinfachungen wirklich helfen.

Die mangelhafte Datenlage macht koordinierte Maßnahmen zur **Korpusentwicklung** erforderlich. Domänenübergreifende Datensätze auf Basis der **DIN SPEC 33429** können die thematische Einseitigkeit überwinden und die Vermischung verschiedener Sprachstufen verhindern. Zusammenarbeit zwischen Forschung, Bildungseinrichtungen und Selbstvertretungsinitiativen sowie synthetische Verfahren können die Datenqualität deutlich steigern.

Statt starrer Vereinfachungen braucht es **adaptive Systeme**, die sich dynamisch an unterschiedliche Bedürfnisse anpassen. Die in dieser Arbeit integrierte **Markierungsfunktion** für schwierige Wörter bietet hierfür eine Ausgangsbasis zur Steuerung individueller Vereinfachungstiefe. Durch kontinuierliches Nutzerfeedback könnten solche Systeme Textverständlichkeit und Lernfortschritt gleichermaßen unterstützen.

Ob leistungsfähige Leichte-Sprache-KI zukünftig jemals auf privaten Endgeräten verfügbar sein wird, bleibt fraglich. Leichte Sprache ist eine kommerzielle Nische – spezialisierte, lokal nutzbare Modelle lassen sich kaum wirtschaftlich entwickeln. Momentan bieten Custom GPTs die einzige praktikable Lösung: erschwinglich, integrierbar und mit brauchbaren Ergebnissen. Auch wenn sie keine perfekte Vereinfachung leisten, sind sie deutlich besser als gar keine Hilfe. Wünschenswert wäre, dass künftige Smartphone-Assistenten auch Leichte Sprache individuell unterstützen.

> Letztendlich zeigt sich: Ohne gesellschaftlichen und rechtlichen Wandel bleiben technische Fortschritte wirkungslos. Verbindliche Barrierefreiheits-Standards, nachhaltige Finanzierung von Selbstvertretung und die Einbindung Betroffener als Co-Entwickler sind Voraussetzungen dafür, dass KI-gestützte Leichte Sprache jenen Menschen zugänglich wird, die sie am dringendsten benötigen.

Literaturverzeichnis

[Alv23] Alves, Caion: European standards for making information easy to read and understand (2023), URL https://easy-to-read.eu/european-standards/, abgerufen am 12. Mai 2025

[AM20] Alva-Manchego, Fernando; Scarton, Carolina und Specia, Lucia: Data-Driven Sentence Simplification: Survey and Benchmark. *Computational Linguistics* (2020), Bd. 46(1): S. 135–187, URL https://aclanthology.org/2020.cl-1.4/

[Ans24] Ansar, Wazib; Goswami, Saptarsi und Chakrabarti, Amlan: A Survey on Transformers in NLP with Focus on Efficiency (2024), URL https://arxiv.org/abs/2406.16893

[Aum22] Aumiller, Dennis und Gertz, Michael: Klexikon: A German Dataset for Joint Summarization and Simplification, in: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, European Language Resources Association, Marseille, France, S. 2693–2701, URL https://aclanthology.org/2022.lrec-1.288/

[BIK25] BIK BITV-Test: BIK BITV-Test: Prüfverfahren WCAG 2.2 Web (2025), URL https://bitvtest.de/pruefverfahren/wcag-22-web, abgerufen am 30. Juni 2025

[Bud20] Buddeberg, Klaus und Grotlüschen, Anke: *LEO 2018: Leben mit geringer Literalität (Buch)*, wbv Media GmbH & Company KG, Bielefeld (2020)

[Bun49] Bundesministerium der Justiz: Grundgesetz für die Bundesrepublik Deutschland (1949), URL https://www.gesetze-im-internet.de/gg/art_3.html, abgerufen am 12. Mai 2025

[Bun02] Bundesministerium der Justiz (Hrsg.): Gesetz zur Gleichstellung von Menschen mit Behinderungen (Behindertengleichstellungsgesetz – BGG) (2002), URL https://www.gesetze-im-internet.de/bgg/BJNR146800002.html, bGBl. I S. 1467; zuletzt geändert am 23.05.2022 durch Art. 7 G (BGBl. I S. 760)

[Bun11] Bundesministerium des Innern: Barrierefreie-Informationstechnik-Verordnung (BITV 2.0) (2011), URL https://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html

[Bun14] Bundeszentrale für politische Bildung: Die Bundes-Zentrale für politische Bildung stellt sich vor (2014), URL https://www.bpb.de/d

ie-bpb/ueber-uns/informationen-in-leichter-sprache/185759/die-bundes-zentrale-fuer-politische-bildung-stellt-sich-vor/, veröffentlicht am 05.06.2014, abgerufen am 4. Juli 2025

[Bun21] Bundesrepublik Deutschland: Gesetz zur Stärkung der Barrierefreiheit (Barrierefreiheitsstärkungsgesetz - BFSG) (2021), URL https://www.gesetze-im-internet.de/bfsg/, bGBl. I S. 2970

[Bun25a] Bundesministerium für Arbeit und Soziales: Arbeits-Recht. Informationen für Arbeit-Nehmer (2025), URL https://www.bmas.de/SharedDocs/Downloads/DE/Publikationen/a711l-arbeits-recht-informationen-fuer-arbeit-nehmer-ls.pdf?__blob=publicationFile&v=1, heft, Artikelnummer A711L, Stand: Januar 2025, abgerufen am 04.07.2025

[Bun25b] Bundesministerium für Arbeit und Soziales: Der Mindest-Lohn. Alle wichtigen Informationen zum Mindest-Lohn (2025), URL https://www.bmas.de/SharedDocs/Downloads/DE/Publikationen/a640l-ml-broschuere-ls.pdf?__blob=publicationFile&v=10, heft, Artikelnummer A640L, Stand: Januar 2025, abgerufen am 04.07.2025

[Cam23] Campbell, Alastair; Adams, Chuck; Montgomery, Rachael Bradley; Cooper, Michael und Kirkpatrick, Andrew: Web Content Accessibility Guidelines (WCAG) 2.2, https://www.w3.org/TR/2023/REC-WCAG22-20231005/ (2023), w3C Recommendation. abgerufen am 15. Juni 2025

[cap] capito – Übersetzungen in Leichte Sprache: capito Text-Check – Verständlichkeitsprüfung nach dem capito-Prinzip, URL https://app.capito.ai/de, abgerufen am 20. Juni 2025

[CFS25] CFS Consulting, Franchise und Sales GmbH: capito (2025), URL https://www.capito.eu/, abgerufen am 12. Juni 2025

[Chu22] Chung, Hyung Won; Hou, Le; Longpre, Shayne; Zoph, Barret; Tay, Yi; Fedus, William; Li, Yunxuan; Wang, Xuezhi; Dehghani, Mostafa; Brahma, Siddhartha; Webson, Albert; Gu, Shixiang Shane; Dai, Zhuyun; Suzgun, Mirac; Chen, Xinyun; Chowdhery, Aakanksha; Castro-Ros, Alex; Pellat, Marie; Robinson, Kevin; Valter, Dasha; Narang, Sharan; Mishra, Gaurav; Yu, Adams; Zhao, Vincent; Huang, Yanping; Dai, Andrew; Yu, Hongkun; Petrov, Slav; Chi, Ed H.; Dean, Jeff; Devlin, Jacob; Roberts, Adam; Zhou, Denny; Le, Quoc V. und Wei, Jason: Scaling Instruction-Finetuned Language Models (2022), URL https://arxiv.org/abs/2210.11416

[Com25] Community Builder: Optimeil: Leichte Sprache Prüfer (2025), URL https://chatgpt.com/g/g-39SG7xnib-optimeil-leichte-sprache-prufer, abgerufen am 5. Juli 2025

[Coo18] Cooper, Michael; Kirkpatrick, Andrew et al.: Web Content Accessibility Guidelines (WCAG) 2.1, https://www.w3.org/TR/WCAG21/ (2018), w3C Recommendation. abgerufen am 15. Juni 2025

[Dad21] Dadu, Tanvi; Pant, Kartikey; Nagar, Seema; Barbhuiya, Ferdous Ahmed und Dey, Kuntal: Text Simplification for Comprehension-based Question-Answering (2021), URL https://arxiv.org/abs/2109.13984

[Dev19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton und Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019), URL https://arxiv.org/abs/1810.04805

[Dig25] DigitalAgentur Brandenburg GmbH: LeiSA (2025), URL https://www.digital-agentur.de/schwerpunkte/mit-kuenstlicher-intelligenz-mehrwerte-schaffen/leichte-sprache-assistent/, abgerufen am 12. Juni 2025

[DIN25] DIN e.V.: DIN SPEC 33429:2025-03 – Empfehlungen zur Leichten Sprache, https://www.dinmedia.de/de/technische-regel/din-spec-33429/387728031 (2025), abgerufen am 15. Juni 2025

[dIufH24] des Innern und für Heimat, Bundesministerium: Web Content Accessibility Guidelines 2.1 (WCAG 2.1) (2024), URL https://www.barrierefreiheit-dienstekonsolidierung.bund.de/Webs/PB/DE/gesetze-und-richtlinien/wcag/wcag-node.html, abgerufen am 15. Juni 2025

[Eur16] Europäische 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 (2016), URL https://eur-lex.europa.eu/legal-content/DE/TXT/HTML/?uri=CELEX:32016L2102&from=DE

[Eur19] Europäische Union: Richtlinie (EU) 2019/882 des Europäischen Parlaments und des Rates vom 17. April 2019 über die Barrierefreiheitsanforderungen für Produkte und Dienstleistungen (2019), URL https://eur-lex.europa.eu/eli/dir/2019/882/oj/deu, usr_lan: DE

[Eur24] Europäisches Parlament und Rat der Europäischen Union: VERORDNUNG (EU) 2024/1689 DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 13. Juni 2024 zur Festlegung harmonisierter Vorschriften für künstliche Intelligenz (2024), URL https://eur-lex.europa.eu/legal-content/DE/TXT/HTML/?uri=OJ:L_202401689

[fAuS25] für Arbeit und Soziales, Bundesministerium: Einheitliche Empfehlungen für Leichte Sprache (2025), URL https://www.bmas.de/DE/Service/Presse/Meldungen/2025/einheitliche-empfehlungen-leichte-sprache.html, abgerufen am 15. Juni 2025

[FH 25] FH Aachen: ErLeSen (2025), URL https://www.fh-aachen.de/forschung/forschungsprojekte/projekte/erlesen, abgerufen am

12. Juni 2025

[Fu23] Fu, Zihao; Lam, Wai; Yu, Qian; So, Anthony Man-Cho; Hu, Shengding; Liu, Zhiyuan und Collier, Nigel: Decoder-Only or Encoder-Decoder? Interpreting Language Model as a Regularized Encoder-Decoder (2023), URL https://arxiv.org/abs/2304.04052

[Gri19] Grigoleit, Sonja: Natural Language Processing. *Europäische Sicherheit & Technik : ES & T* (2019), Bd. 68(4): S. 69, URL https://publica.fraunhofer.de/handle/publica/260578

[Gro19] Grotlüschen, Anke; Buddeberg, Klaus; Dutz, Gregor; Heilmann, Lisanne und Stammer, Christopher: LEO 2018 – Leben mit geringer Literalität (Pressebroschüre), Pressebroschüre, Universität Hamburg, Hamburg (2019), URL https://leo.blogs.uni-hamburg.de/wp-content/uploads/2022/09/LEO2018-Presseheft.pdf, abgerufen am 15. April 2025

[HahoJ] Hahn, Martin: Selbstbestimmung – das Thema der 90er Jahre, Techn. Ber., Bundesvereinigung Lebenshilfe (o.J.), URL https://www.lebenshilfe.de/fileadmin/Redaktion/Bilder/LandingPages/Geschichte/Downloads/90Hahn-Selbstbestimmung.pdf, abgerufen am 20. Juni 2025

[Hal25] Haller, Pascal: Leichte Sprache Übersetzer (2025), URL https://www.leichte-sprache-uebersetzer.de/, abgerufen am 12. Juni 2025

[Hoc25] Hochschule Karlsruhe: STARK-LS (2025), URL https://www.h-ka.de/ilin/projekte/stark-ls, abgerufen am 12. Juni 2025

[Hug23] Hugging Face: Hugging Face - The AI community building the future., https://huggingface.co/ (2023), abgerufen am 30.06.2025

[Inc17] Inclusion Europe: *Informationen für alle: Europäische Regeln, wie man Informationen leicht lesbar und leicht verständlich macht*, Eigenverlag, Brüssel (2017), URL https://www.inclusion-europe.eu/wp-content/uploads/2017/06/DE_Information_for_all.pdf

[Inc25] Inclusion Europe: Inclusion Europe – Easy to Read (2025), URL https://www.inclusion-europe.eu/easy-to-read, abgerufen am 03. Juni 2025

[Kel14] Kellermann, Gudrun: Leichte und Einfache Sprache – Versuch einer Definition (2014), URL https://www.bpb.de/shop/zeitschriften/apuz/179341/leichte-und-einfache-sprache-versuch-einer-definition/, abgerufen am 14. Juni 2025

[Klö24] Klöser, Lars; Beele, Mika; Schagen, Jan-Niklas und Kraft, Bodo: German Text Simplification: Finetuning Large Language Models with Semi-Synthetic Data (2024), URL https://arxiv.org/abs/2402.10675, arXiv:2402.10675 [cs.CL]

[Kre25] Kreuzer, Michael: Custom GPT: Leichte Sprache zu schwere Sprache (2025), URL https://chatgpt.com/g/g-6867bf2e3c3081919624cb29cc52a73a-leichte-sprache-zu-schwere-sprache, abgerufen am 4. Juli 2025

[Leb25] Lebenshilfe e.V.: Lebenshilfe – Leichte Sprache (2025), URL https://www.leichte-sprache.de, abgerufen am 03. Juni 2025

[Maa15] Maass, Christiane: *Leichte Sprache. Das Regelbuch*, Bd. 1 von *Barrierefreie Kommunikation*, LIT Verlag Dr. W. Hopf, Berlin (2015)

[Maa16] Maass, Christiane und Bredel, Ursula: *Ratgeber Leichte Sprache. Die wichtigsten Regeln und Empfehlungen für die Praxis*, Duden, Berlin (2016)

[Mad23] Madina, Margot; Gonzalez-Dios, Itziar und Siegel, Melanie: Easy-to-Read in Germany: A Survey on its Current State and Available Resources (2023), URL https://arxiv.org/abs/2306.03189

[Mena] Mensch zuerst – Netzwerk People First Deutschland e.V.: Verein – Mensch zuerst, URL https://www.menschzuerst.de/wer-sind-wir/verein/, abgerufen am 20. Juni 2025

[Menb] Mensch zuerst – Netzwerk People First Deutschland e.V.: Was tun wir – Mensch zuerst, URL https://www.menschzuerst.de/was-tun-wir/, abgerufen am 20. Juni 2025

[Nad19] Naderi, Babak; Mohtaj, Salar; Ensikat, Kaspar und Möller, Sebastian: Subjective Assessment of Text Complexity: A Dataset for German Language. *arXiv preprint arXiv:1904.07733* (2019)

[Neta] Netzwerk Leichte Sprache e.V.: Die Arbeit vom Netzwerk, URL https://www.netzwerk-leichte-sprache.de/ls/das-netzwerk/die-arbeit-vom-netzwerk, abgerufen am 12. Mai 2025

[Netb] Netzwerk Leichte Sprache e.V.: Mitglieder vom Verein, URL https://www.netzwerk-leichte-sprache.de/ls/das-netzwerk/mitglieder-vom-verein, abgerufen am 10. Mai 2025

[Netc] Netzwerk Leichte Sprache e.V.: Was ist das Netzwerk?, URL https://www.netzwerk-leichte-sprache.de/ls/das-netzwerk/was-ist-das-netzwerk, abgerufen am 10. Mai 2025

[Net14] Netzwerk Leichte Sprache: *Leichte Sprache. Ein Ratgeber*, BMAS, Hausdruckerei, Bonn, unveränderter nachdruck 2022 Aufl. (2014), URL https://www.gemeinsam-einfach-machen.de/SharedDocs/Downloads/DE/AS/UN_BRK/LS_EinRatgeber.pdf?__blob=publicationFile&v=4

[Net21] Netzwerk Leichte Sprache: *LEICHTE SPRACHE verstehen: Mit Textbeispielen aus dem Alltag und Tipps für die Praxis in leichter Sprache*, marixverlag (2021), URL https://books.google.de/books?id=4BUxEAAAQBAJ

[Net22] Netzwerk Leichte Sprache: Die Regeln für Leichte Sprache, Techn. Ber., Netzwerk Leichte Sprache e.V., Berlin (2022), URL https://www.netzwerk-leichte-sprache.de/fileadmin/content/documents/regeln/Regelwerk_NLS_Neuauflage-2022.pdf, abgerufen am 16 April 2025

[Net25] Netzwerk Leichte Sprache: Netzwerk Leichte Sprache (2025), URL https://www.netzwerk-leichte-sprache.de, abgerufen am 03. Juni 2025

[Neu25] Neubauer, Mansour: Was ist Einfache Sprache? Welche Prinzipien und Regeln folgt sie? (2025), URL https://www.einfache-sprache.com/definition.html, abgerufen am 14. Juni 2025

[OEC24] OECD: Empfehlung des Rates zu künstlicher Intelligenz, Techn. Ber. OECD/LEGAL/0449, Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (OECD), Paris (2024), URL https://legalinstruments.oecd.org/api/download/?uri=/public/2d194c24-d0be-4e7f-9736-1fffa3324ad6.pdf

[Paa20] Paass, Gerhard und Hecker, Dirk: *Künstliche Intelligenz : Was steckt hinter der Technologie der Zukunft?*, Springer Fachmedien Wiesbaden, Wiesbaden, 1st ed. 2020 Aufl. (2020), URL https://doi.org/10.1007/978-3-658-30211-5

[pep25] pepper – Büro für Grafik- und Webdesign: Lesbarkeit mit dem Flesch-Reading-Ease analysieren (2025), URL https://barrierefreies.design/werkzeuge/lesbarkeit-analysieren, abgerufen am 5. Juli 2025

[Rad18] Radford, Alec und Narasimhan, Karthik: Improving Language Understanding by Generative Pre-Training (2018), URL https://gwern.net/doc/www/s3-us-west-2.amazonaws.com/d73fdc5ffa8627bce44dcda2fc012da638ffb158.pdf

[Rad25] Radzimanowski, Maria und Genz, Lukas: KI-Prototyp Leichte-Sprache-Assistent (LeiSA) (2025), abgerufen am 23 Juni 2025

[Reg13] Regeringskansliet: Lättläst – en översyn av den statliga stödstrukturen för lättläst, Techn. Ber. SOU 2013:58, Schwedische Regierung, Stockholm (2013), URL https://www.regeringen.se/contentassets/1da22c54e6d74ce59fcdddac74936f26/lattlast-hela-dokumentet-sou-201358//

[Säu20] Säuberli, Andreas; Ebling, Sarah und Volk, Martin: Benchmarking Data-driven Automatic Text Simplification for German, in: Núria Gala und Rodrigo Wilkens (Herausgeber) *Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)*, European Language Resources Association, Marseille, France, S. 41–48, URL https://aclanthology.org/2020.readi-1.7/

[Tau22] Taulli, Tom: *Grundlagen der Künstlichen Intelligenz : Eine nichttechnische Einführung*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. 2022 Aufl. (2022), URL https://doi.org/10.1007/978-3-662-66283-0

[Tob23] Toborek, Vanessa; Busch, Moritz; Bossert, Malte; Bauckhage, Christian und Welke, Pascal: A New Aligned Simple German Corpus, in: *Proceedings of ACL 2023 (Volume 1: Long Papers)*, S. 11393–11412, URL https://aclanthology.org/2023.acl-long.638

[TUE16] TUEV SUED Product Service GmbH und H&H Communication Lab: Erklaerung des Verfahrens TUEV geprufte Verstaendlichkeit (2016), URL https://www.service-tested.de/wp-content/uploads/2016/04/Erkl%C3%A4rung-Verfahren-T%C3%9CV-gepr%C3%BCfte-Verst%C3%A4ndlichkeit.pdf, abgerufen am 20. Juni 2025

[Uni06] United Nations: Article 2 – Definitions (2006), URL https://social.desa.un.org/issues/disability/crpd/article-2-definitions, abgerufen am 12. Mai 2025

[Uni25] Universität Hildesheim: KI-GesKom (2025), URL https://www.uni-hildesheim.de/leichtesprache/forschung-und-projekte/projekte/ki-geskom/, abgerufen am 12. Juni 2025

[Vas23] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz und Polosukhin, Illia: Attention Is All You Need (2023), URL https://arxiv.org/abs/1706.03762

[VER25] VERSO Dresden gGmbH: KI-Tools und Leichte Sprache. Eine Untersuchung KI-generierter Texte anhand der DIN SPEC 33429 (2025), URL https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-964271, angenommene Version / Postprint / Autorenversion, 2025-06-23

[Wor] Wortliga GmbH: Wortliga Textanalyse – Online-Tool zur Verständlichkeitsprüfung, URL https://wortliga.de/textanalyse/, abgerufen am 20. Juni 2025

[WOR25] WORTLIGA Tools GmbH: Wortliga (2025), URL https://wortliga.de/texte-umschreiben/, abgerufen am 12. Juni 2025

[Xu16] Xu, Wei; Napoles, Courtney; Pavlick, Ellie; Chen, Quanze und Callison-Burch, Chris: Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics* (2016), Bd. 4: S. 401–415, URL https://www.aclweb.org/anthology/Q16-1029

[Xue21] Xue, Linting; Constant, Noah; Roberts, Adam; Kale, Mihir; Al-Rfou, Rami; Siddhant, Aditya; Barua, Aditya und Raffel, Colin: mT5: A massively multilingual pre-trained text-to-text transformer (2021), URL https://arxiv.org/abs/2010.11934

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

.

A Anhang

A.1 Hintergrund

Tabelle A.1: Potenzielle Zielgruppen für Leichte Sprache in Deutschland

Gruppe	**Schätzung**	**Quelle**
Menschen mit Legasthenie	0,47 Mio.	[Maa16]
Menschen mit geistiger Behinderung	0,4–0,8 Mio.	[Maa16]
Menschen mit Demenz	1,3 Mio.	[Maa16]
Prälingualhörgeschädigte	0,08 Mio.	[Maa16]
Menschen mit Aphasie	0,13–0,24 Mio.	[Maa16]
Funktionale Analphabeten (Alpha-Level 1–3)	6,2 Mio.	[Gro19]
Menschen mit Deutsch als Zweitsprache	>1 Mio.	[Maa16]
Fehlerhaftes Schreiben (Alpha-Level 4)	10,6 Mio.	[Gro19]
Gesamtschätzung (nicht bereinigt)	**ca. 20 Mio.**	Eigene Berechnung auf Basis der Quellen

Tabelle A.2: Übersicht der Regelwerke für Leichte Sprache

Regelwerk	**Anzahl Regeln**	**Anzahl Kategorien**	**Struktur**
Inclusion Europe [Inc17]	132	6	• Allgemeine Regeln (20) • Geschriebene Informationen (44) • Elektronische Informationen (29) • Video-Informationen (22) • Audio-Informationen (17)
Netzwerk Leichte Sprache [Net22]	47	6	• Wörter (12) • Zahlen/Zeichen (9) • Sätze (4) • Texte (4) • Gestaltung/Bilder (17) • Prüfen (1)
BITV 2.0 [Bun11]	13	–	• Kompakte Sammlung • Mehrere Aspekte pro Regel

Tabelle A.3: Vergleich von BITV 2.0 (2022) [Bun11] und dem Regelwerk des Netzwerks Leichte Sprache (2022, im Folgenden: NLS) [Net22]

BITV-Regel (Anlage 2)	**Regel NLS**	**Übereinstimmung**
Keine Abkürzungen, Silbentrennung am Zeilenende, Verneinungen, Konjunktiv, Passiv, Genitiv	W6, W8, W9, W10, W11, G6	Ja
Persönliche Ansprache	T1	Ja
Gleiche Begriffe verwenden	W4	Ja
Einfache und kurze Wörter, Erklärung von Fremdwörtern, Bindestrich bei langen zusammengesetzten Wörtern	W1, W3, W5	Ja
Kurze, klar gegliederte Sätze	S1, S2	Ja
Keine Sonderzeichen und Klammern	Z9	Ja
Absätze, Überschriften, Listen	G9, G11	Ja
Wichtige Inhalte zuerst	T4	Ja
Klare Schriftart, große Schrift, max. 2 Schriftarten, Hervorhebungen	G1, G2, G11	**Teilweise (max. 1 Schriftart)**
Linksbündig, Satz pro Zeil, heller Hintergrund	G4, G5, G12	Ja
Symbole und Bilder unterstützend einsetzen	G15–G17	Ja
Keine Fließtexte bei Anschriften	G10	Ja
Übersichtliche Tabellen	–	**Nein**

Tabelle A.4: Quantitative Bewertung der drei Tools nach DIN SPEC 33429 [VER25]

Kriterium	**Alle 3 Tools**	**Bewertung**
Wortlänge	4–5 Zeichen (kurze Wörter)	**Erfüllt**
Satzlänge	6–7 Token (kurze Sätze)	**Erfüllt**
Grundwortschatz	48–54% (hoher Anteil)	**Erfüllt**
Nominal-/Verbalstil	57–65% Nominalstil	**Nicht erfüllt**
Passiv-Vermeidung	0–2 Passivformen	**Teilweise erfüllt (erfüllt: Capito)**
Konjunktiv-Vermeidung	1–2 Konjunktivformen	**Nicht erfüllt**

Tabelle A.5: Qualitative Bewertung der drei Tools nach DIN SPEC 33429 [VER25]

Kriterium	**Capito**	**Wortliga**	**Leichte Sprache Übersetzer**	**Bewertung**
WORTEBENE				
Zentraler Wortschatz	Nicht erfüllt	Erfüllt	Nicht erfüllt	Teilweise erfüllt (erfüllt: Wortliga)
Kurze Wörter	Nicht erfüllt	Erfüllt	Nicht erfüllt	Teilweise erfüllt (erfüllt: Wortliga)
Zerlegung langer Wörter	Nicht erfüllt	Nicht erfüllt	Erfüllt	Teilweise erfüllt (erfüllt: Übersetzer)
Erläuterung schwieriger Wörter	Nicht erfüllt	Nicht erfüllt	Nicht erfüllt	Nicht erfüllt
Keine Synonyme	Erfüllt	Erfüllt	Erfüllt	Erfüllt
Keine Metaphern	Nicht erfüllt	Erfüllt	Erfüllt	Teilweise erfüllt (erfüllt: Wortliga, Übersetzer)
SATZEBENE				
Kurze Sätze	Erfüllt	Nicht erfüllt	Erfüllt	Teilweise erfüllt (erfüllt: Capito, Übersetzer)
Einfache Satzstruktur	Erfüllt	Nicht erfüllt	Erfüllt	Teilweise erfüllt (erfüllt: Capito, Übersetzer)
Sinneinheiten pro Zeile	Erfüllt	Nicht erfüllt	Erfüllt	Teilweise erfüllt (erfüllt: Capito, Übersetzer)
Verbalstil	Erfüllt	Erfüllt	Erfüllt	Erfüllt
GRAMMATIK				
Aktiv statt Passiv	Erfüllt	Erfüllt	Erfüllt	Erfüllt
Indikativ statt Konjunktiv	Erfüllt	Erfüllt	Erfüllt	Erfüllt
Wenig Zeitformen	Nicht erfüllt	Nicht erfüllt	Nicht erfüllt	Nicht erfüllt
STRUKTUR & INHALT				
Textgliederung	Erfüllt	Erfüllt	Erfüllt	Erfüllt
Sinnerhaltung	Nicht erfüllt	Nicht erfüllt	Erfüllt	Teilweise erfüllt (erfüllt: Übersetzer)

A.2 Konzept

Tabelle A.6: WCAG 2.2 Prüfschritte und ihre Relevanz [BIK25]

Prüfschritt	Relevanz	Details
1.1.1a,b – Alternativetexte für Bedienelemente, Grafiken und Objekte	hoch	Alt-Texte für interaktive Elemente und informative Grafiken
1.1.1c – Leere alt-Attribute für Layoutgrafiken	nicht relevant	Layoutgrafiken mit leerem alt-Attribut – bei minimalistischer Gestaltung irrelevant
1.1.1d – Alternativen für CAPTCHAs	nicht relevant	CAPTCHA wird nicht verwendet
1.2.1 – Alternativen für Audiodateien und stumme Videos	nicht relevant	Keine Audiodateien oder stumme Videos
1.2.2 – Aufgezeichnete Videos mit Untertiteln	nicht relevant	Keine Videos
1.2.3 – Audiodeskription oder Volltext-Alternative für Videos	nicht relevant	Keine Videos mit Audiodeskription notwendig
1.2.4 – Videos (live) mit Untertiteln	nicht relevant	Kein Livestream
1.2.5 – Audiodeskription für Videos	nicht relevant	Keine Audiodeskription nötig
1.3.1a – HTML-Strukturelemente für Überschriften	hoch	Überschriften mit h1-h6 strukturiert
1.3.1b – HTML-Strukturelemente für Listen	hoch	Listenstruktur verwenden
1.3.1c – HTML-Strukturelemente für Zitate	nicht relevant	Zitate kaum relevant
1.3.1d – Inhalt gegliedert	hoch	Klar gegliederter Seiteninhalt
1.3.1e – Datentabellen richtig aufgebaut	nicht relevant	Keine Datentabellen geplant
1.3.1f – Zuordnung von Tabellenzellen	nicht relevant	Keine Tabellenzellen
1.3.1g – Kein Strukturmarkup für Layouttabellen	nicht relevant	Keine Layouttabellen
1.3.1h – Beschriftung von Formularelementen programmatisch ermittelbar	hoch	Formularfelder beschriftet
1.3.2 – Sinnvolle Reihenfolge	hoch	Fokusreihenfolge folgt visueller Logik
1.3.3 – Ohne Bezug auf sensorische Merkmale nutzbar	hoch	Informationen nicht nur farblich vermittelt
1.3.4 – Keine Beschränkung der Bildschirmausrichtung	hoch	Bildschirmdrehung nicht eingeschränkt

weiter auf der nächsten Seite

Prüfschritt	**Relevanz**	**Details**
1.3.5 – Eingabefelder zu Nutzerdaten vermitteln den Zweck	nicht relevant	Keine personenbezogenen Datenabfragen
1.4.1 – Ohne Farben nutzbar	hoch	Farbunabhängige Erkennung bei Buttons
1.4.2 – Ton abschaltbar	nicht relevant	Kein Ton vorhanden
1.4.3 – Kontraste von Texten ausreichend	hoch	Textkontrast mind. 4.5:1
1.4.4 – Text auf 200 % vergrößerbar	hoch	Text muss auf 200 % vergrößerbar sein
1.4.5 – Schriftgrafiken	nicht relevant	Keine Schriftgrafiken
1.4.10 – Inhalte brechen um	hoch	Responsives Design
1.4.11 – Kontraste von Grafiken und grafischen Bedienelementen ausreichend	hoch	Buttons/Symbole mit 3:1 Kontrast
1.4.12 – Textabstände anpassbar	hoch	Textabstände müssen anpassbar sein
1.4.13 – Eingeblendete Inhalte bedienbar	mittel	Tooltips/Hinweise müssen bedienbar sein
2.1.1 – Ohne Maus nutzbar	hoch	Alles muss per Tastatur bedienbar sein
2.1.2 – Keine Tastaturfalle	hoch	Fokus darf nicht hängen bleiben
2.1.4 – Tastatur-Kurzbefehle abschaltbar oder anpassbar	niedrig	Keine komplexen Tastaturkürzel geplant
2.2.1 – Zeitbegrenzungen anpassbar	nicht relevant	Keine Zeitbegrenzung vorhanden
2.2.2 – Bewegte Inhalte abschaltbar	nicht relevant	Keine Bewegung vorhanden
2.3.1 – Verzicht auf Flackern	niedrig	Keine flackernden Inhalte
2.4.1 – Bereiche überspringbar	hoch	Sprunglink, HTML5 vorhanden
2.4.2 – Sinnvolle Dokumenttitel	mittel	Titel der Webseite im title-Tag eindeutig
2.4.3 – Schlüssige Reihenfolge bei der Tastaturbedienung	hoch	Tab-Reihenfolge muss logisch sein
2.4.4 – Aussagekräftige Linktexte	mittel	Linktexte müssen verständlich sein
2.4.5 – Alternative Zugangswege	mittel	Zwei alternative Zugangswege (z. B. Suche)
2.4.6 – Aussagekräftige Überschriften und Beschriftungen	hoch	Überschrift oder Label sinnvoll
2.4.7 – Aktuelle Position des Fokus deutlich	hoch	Fokus visuell sichtbar
2.4.11 – Fokus nicht verdeckt (Minimum)	hoch	Fokus nie außerhalb des sichtbaren Bereichs
2.5.1 – Alternativen für komplexe Zeiger-Gesten	nicht relevant	Keine Gestensteuerung

weiter auf der nächsten Seite

Prüfschritt	**Relevanz**	**Details**
2.5.2 – Zeigergesten-Eingaben können abgebrochen oder widerrufen werden	nicht relevant	Keine Gestenbedienung
2.5.3 – Sichtbare Beschriftung Teil des zugänglichen Namens	hoch	Visueller Text = technisch zugänglicher Name
2.5.4 – Alternativen für Bewegungsaktivierung	nicht relevant	Keine Bewegungssteuerung
2.5.7 – Ziehbewegungen	nicht relevant	Kein Drag & Drop
2.5.8 – Zielgröße (Minimum)	hoch	Buttons mind. 24×24 px
3.1.1 – Hauptsprache angegeben	mittel	HTML-Tag html lang="de"gesetzt
3.1.2 – Anderssprachige Wörter und Abschnitte ausgezeichnet	mittel	Fremdwörter haben lang="..."
3.2.1 – Keine unerwartete Kontextänderung bei Fokus	hoch	Fokus verändert keine Inhalte automatisch
3.2.2 – Keine unerwartete Kontextänderung bei Eingabe	hoch	Eingaben verändern keine Inhalte automatisch
3.2.3 – Konsistente Navigation	hoch	Navigation ist überall gleich
3.2.4 – Konsistente Bezeichnung	hoch	Gleiche Begriffe für gleiche Funktionen
3.2.6 – Konsistente Hilfe	nicht relevant	Hilfeoptionen überall gleich umgesetzt
3.3.1 – Fehlererkennung	hoch	Fehlermeldung bei leerem Feld
3.3.2 – Beschriftungen von Formularelementen vorhanden	hoch	Eingabefeld sichtbar beschriftet
3.3.3 – Hilfe bei Fehlern	hoch	z. B. „Feld darf nicht leer sein"
3.3.4 – Fehlervermeidung wird unterstützt	hoch	Validierung vor Absenden
3.3.7 – Redundanter Eintrag	nicht relevant	Kein mehrfaches Ausfüllen nötig
3.3.8 – Zugängliche Authentifizierung (Minimum)	nicht relevant	Kein Login erforderlich
4.1.1 – Korrekte Syntax	niedrig	Sauberer HTML-/ARIA-Code
4.1.2 – Name, Rolle, Wert verfügbar	hoch	Buttons und Felder technisch korrekt ausgezeichnet
4.1.3 – Statusmeldungen programmatisch verfügbar	hoch	Statusmeldungen für Screenreader per aria-live

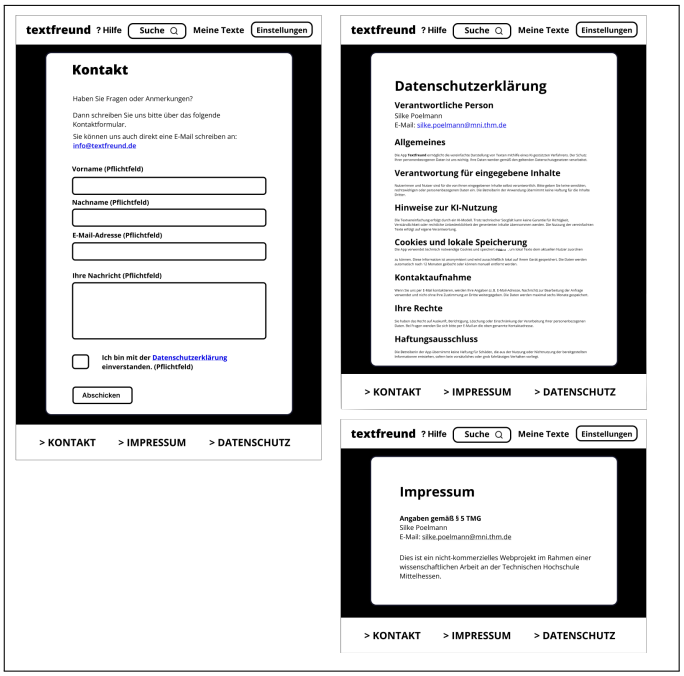


Abbildung A.1: Mockups der statischen Seiten: Kontakt, Impressum und Datenschutz.

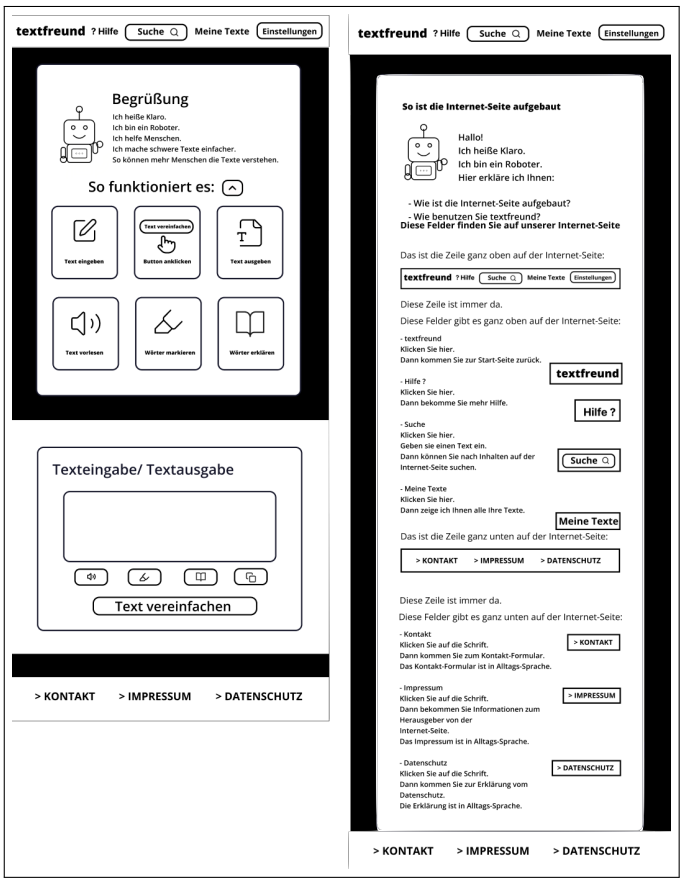


Abbildung A.2: Mockups der Startseite und Hilfeseite.

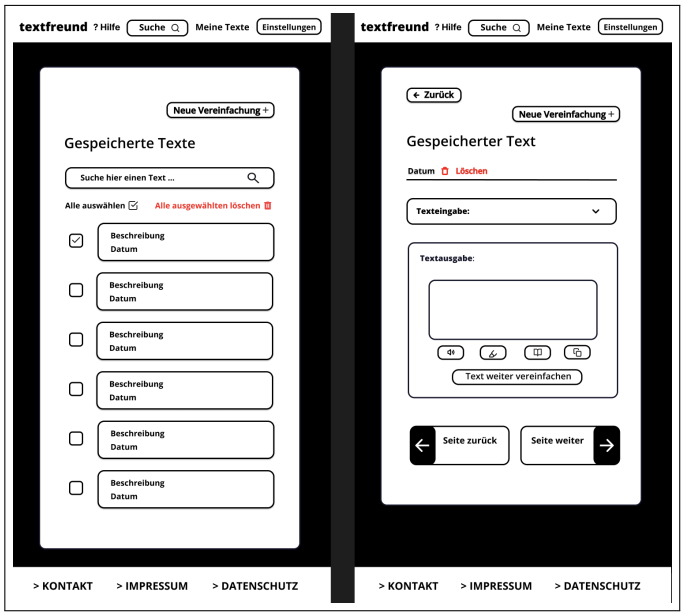


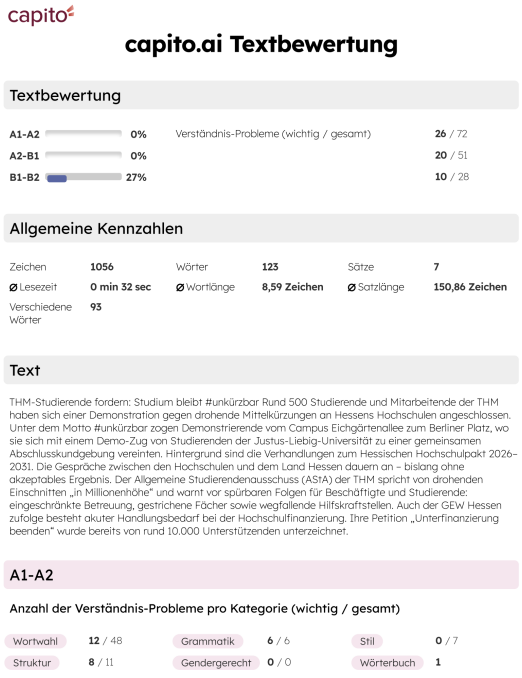
Abbildung A.3: Mockups der Detailseite und Archivseite.

Abbildung A.4: Bewertung des Ausgangstextes mit dem pepper Lesbarkeits-Analyzer.

✅ **Regelprüfung:**

Regel (Regelname)	Erfüllt (Ja/Nein)	Textstelle/n (Erläuterung)
1. Kurze Sätze	❌	Mehrere sehr lange Sätze, z. B. ab „Hintergrund sind die Verhandlungen…“
2. Ein Satz = eine Information	❌	„…zogen Demonstrierende vom Campus…“ enthält mehrere Informationen
3. Schwierige Wörter vermeiden	❌	„Mittelkürzungen“, „Demonstration“, „Verhandlungen“, „akzeptables Ergebnis“
4. Wenn schwere Wörter nötig sind: erklären	❌	Keine Erklärungen zu „THM“, „GEW“, „AStA“ oder „Hochschulpakt“
5. Keine Abkürzungen	❌	„THM“, „GEW“, „AStA“ werden nicht erklärt
6. Keine Fremdwörter	❌	„Petition“, „Demo-Zug“, „Kundgebung“, „Hochschulpakt“ sind Fremdwörter
7. Aktiv schreiben	✅	„Studierende fordern“, „Demonstrierende zogen“ – aktiv formuliert
8. Immer gleiche Wörter für gleiche Dinge	❌	„Demonstration“, „Demo-Zug“, „Kundgebung“ – gleiche Sache, verschiedene Wörter
9. Wörter trennen mit Gedankenstrichen	✅	Keine langen zusammengesetzten Wörter ohne Trennung
10. Keine bildliche Sprache	✅	Keine Metaphern oder Sprichwörter enthalten
11. Keine Ironie oder Witze	✅	Keine Ironie im Text
12. Große Schrift	❌	Nicht beurteilbar, Textformat nicht bekannt
13. Genug Zeilenabstand	❌	Nicht beurteilbar, Format nicht bekannt
14. Leicht lesbare Schrift	❌	Nicht beurteilbar, Format nicht bekannt
15. Bilder helfen	❌	Keine Bilder im Text vorhanden
16. Einfache Grammatik	❌	Viele Nebensätze und verschachtelte Strukturen
17. Keine Genitiv-Formulierungen	✅	Kein Genitiv wie „der Hochschule der THM“
18. Gliederung durch Absätze	✅	Der Text ist in Absätze gegliedert
19. Überschriften helfen	❌	Nur eine Überschrift vorhanden („THM-Studierende fordern…“)
20. Leichte Sprache kennzeichnen	❌	Kein Hinweis, dass es sich um Leichte Sprache handelt

Abbildung A.5: Bewertung des Ausgangstextes mit Optimeil Prüfer.



capito

capito.ai Textbewertung

Textbewertung

A1-A2	0%	Verständnis-Probleme (wichtig / gesamt)	**26** / 72
A2-B1	0%		**20** / 51
B1-B2	27%		**10** / 28

Allgemeine Kennzahlen

Zeichen	**1056**	Wörter	**123**	Sätze	**7**
Ø Lesezeit	**0 min 32 sec**	Ø Wortlänge	**8,59 Zeichen**	Ø Satzlänge	**150,86 Zeichen**
Verschiedene Wörter	**93**				

Text

THM-Studierende fordern: Studium bleibt #unkürzbar Rund 500 Studierende und Mitarbeitende der THM haben sich einer Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen. Unter dem Motto #unkürzbar zogen Demonstrierende vom Campus Eichgärtenallee zum Berliner Platz, wo sie sich mit einem Demo-Zug von Studierenden der Justus-Liebig-Universität zu einer gemeinsamen Abschlusskundgebung vereinten. Hintergrund sind die Verhandlungen zum Hessischen Hochschulpakt 2026–2031. Die Gespräche zwischen den Hochschulen und dem Land Hessen dauern an – bislang ohne akzeptables Ergebnis. Der Allgemeine Studierendenausschuss (AStA) der THM spricht von drohenden Einschnitten „in Millionenhöhe“ und warnt vor spürbaren Folgen für Beschäftigte und Studierende: eingeschränkte Betreuung, gestrichene Fächer sowie wegfallende Hilfskraftstellen. Auch der GEW Hessen zufolge besteht akuter Handlungsbedarf bei der Hochschulfinanzierung. Ihre Petition „Unterfinanzierung beenden“ wurde bereits von rund 10.000 Unterstützenden unterzeichnet.

A1-A2

Anzahl der Verständnis-Probleme pro Kategorie (wichtig / gesamt)

Wortwahl	**12** / 48	Grammatik	**6** / 6	Stil	**0** / 7
Struktur	**8** / 11	Gendergerecht	**0** / 0	Wörterbuch	**1**

Abbildung A.6: Bewertung eines Textes mit capito.

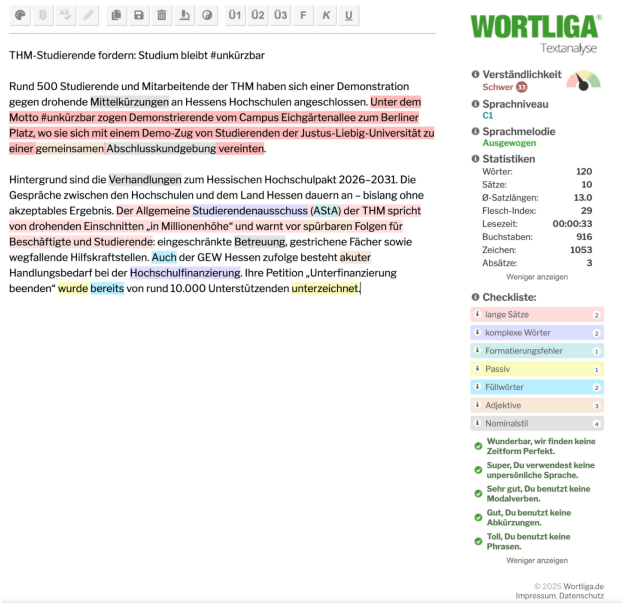


Abbildung A.7: Bewertung eines Textes mit Wortliga.

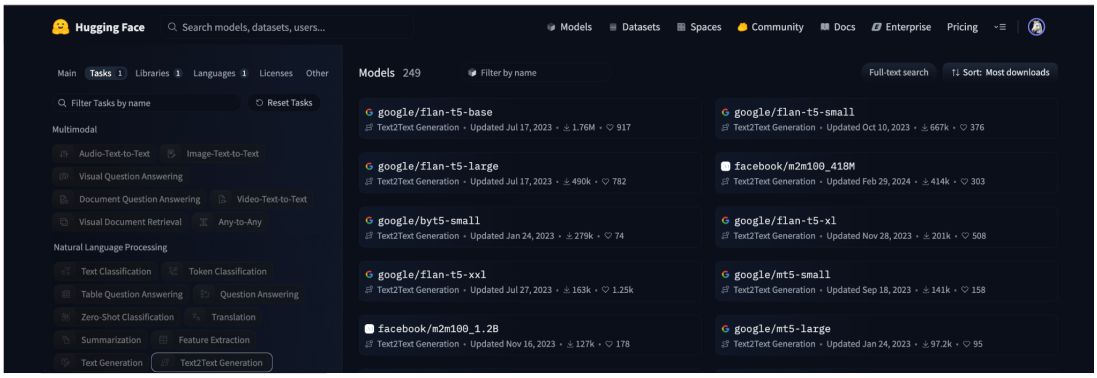


Abbildung A.8: Gefilterte Modellauswahl auf Hugging Face mit den Kriterien *Text2Text-Generation*, *Transformers* und *Deutsch* (249 Treffer).

Vollständige Testergebnisse

mT5 Small:

1. `<extra_id_0>.`
2. `<extra_id_0> an`
3. `<extra_id_0>.`
4. `<extra_id_0>.`

5. <extra_id_0>.
6. <extra_id_0>.
7. <extra_id_0>.

mT5 Base:

1. <extra_id_0> auch in Hessen.
2. <extra_id_0>: Demonstration gegen Mittelk
3. <extra_id_0> - THM -
4. <extra_id_0> - THM Hessen
5. <extra_id_0> Äuto"→ Leicht: Äuto"
6. <extra_id_0>. Suche nach:
7. <extra_id_0> Schritt für Schritt

mT5 Large:

1. <extra_id_0> Mittelkürzungen an Hessens Hochschulen <extra_id_1> 500 Studierende...
2. <extra_id_0> 500 Studierende der THM haben einer Demonstration...
3. <extra_id_0>. Verwende einfache Wörter und kurze Sätze...
4. <extra_id_0>: Verwende keine Fremdwörter und langen Sätze...
5. <extra_id_0>: "Kraftfahrzeug"→ Leicht: Äutoüsw...
6. <extra_id_0> Kürzungen. A) Rund 500 Studenten...
7. <extra_id_0> den Text Schritt für Schritt...

mT5 XL:

1. <extra_id_0> Tag der Hochschule in Wiesbaden. Die THM beteiligt sich nicht an der Demonstration.
2. <extra_id_0> Demonstration in Wiesbaden...
3. <extra_id_0> für Menschen mit Lernschwierigkeiten...
4. <extra_id_0>.de - THM - Technische Hochschule Mittelhessen e.V...

5. <extra_id_0>.de – Hochschule Hessen-Thüringen e.V. (THM)...
6. <extra_id_0> gegen Kürzungen. D) 500 Studenten protestierten...
7. <extra_id_0>.de – Hochschule Hessen-Thüringen e.V...

FLAN-T5 Small:

1. During 500 students and staff of the THM have a demonstration against drohende Mittelkürzungen at Hessen's Hochschules.
2. Rund 500 Studierende und Mitarbeiter der THM haben eine Demonstration gegen drohende Mittelkürzungen in Hessens Hochschulen angeschlossen.
3. Over 500 students and staff of the THM have a demonstration against drohende Mittelkürzungen at Hessen's Hochschules.
4. Over 500 students and staff of the THM have a demonstration against drohende Mittelkürzungen at Hessen's Hochschules.
5. Over 500 students and staff of the THM have a demonstration against drohende Mittelkürzungen at Hessen's Hochschules.
6. A
7. In addition to 500 students and staff of the THM, a Demonstration against drohende Mittelkürzungen in Hessens Hochschulen has been initiated.

FLAN-T5 Base:

1. Rund 500 Studierende und Arbeitnehmer der THM haben eine Demonstration gegen den drohende Mittelkürzungen in Hessens Hochschulen angeschlossen.
2. Rund 500 Studierende und Arbeitnehmer der THM haben eine Demonstration gegen den schädlichen Mittelkürzungen in Hessens Hochschulen eingeschlossen.
3. Rund 500 Studierende und Arbeitnehmer der THM haben eine Demonstration gegen den drohende Mittelkürzungen an Hessens Hochschulen geschlossen.
4. Rund 500 Studierende und Arbeitnehmer der THM haben eine Demonstration gegen den drohende Mittelkürzungen an Hessens Hochschulen geschlossen.
5. Rund 500 Studierende und Arbeitnehmer der THM haben eine Demonstration gegen den drohende Mittelkürzungen an Hessens Hochschulen geschlossen.

6. A) Rund 500 Studierende und Mitarbeitende der THM haben eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen. B) 500 Studenten protestierten gegen Kürzungen. C) Viele Uni-Leute protestierten.

7. Rund 500 Studierende und Arbeitnehmer der THM haben eine Demonstration gegen den drohende Mittelkürzungen in Hessens Hochschulen angeschlossen.

FLAN-T5 Large:

1. Rund 500 Studierende und Mitarbeitende der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

2. Rund 500 Studierende und Mitarbeiter der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

3. Die Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen hat rund 500 Studierende und Mitarbeitende der THM angeschlossen.

4. Unter 500 Studierende und Mitarbeitende der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

5. Kurz: Rund 500 Studierende und Mitarbeitende der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

6. A

7. Die Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen hat rund 500 Studierende und Mitarbeitende der THM zusammengeschlossen.

FLAN-T5 XL:

1. Rund 500 Studierende und Mitarbeiter der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

2. Rund 500 THM-Studenten und Mitarbeiter haben sich für eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

3. Rund 500 Studierende und Mitarbeiter der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

4. Vereinfache das Text für Menschen mit Lernschwäche. Verwende keine Fremdwörter und keine langen Sätze.

5. Rund 500 Studierende und Mitarbeiter der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

6. A) Rund 500 Studierende und Mitarbeitende der THM haben sich einer Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

7. Rund 500 Studierende und Mitarbeiter der THM haben sich eine Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

Leo 7B:

1. Rund 500 Studenten und Mitarbeiter von der THM haben demonstriert.

2. Rund 500 Studenten und Mitarbeiter von der THM haben demonstriert.

3. Rund 5leute haben demonstriert.

4. Rund 500 Studenten und Mitarbeiter von der THM haben demonstriert.

5. Unvollständig - bricht ab

6. A) Rund 500 Studierende und Mitarbeitende der THM haben sich einer Demonstration gegen drohende Mittelkürzungen an Hessens Hochschulen angeschlossen.

7. Unvollständig - bricht ab

Leo 13B:

1. Etwa 500 Studenten und Mitarbeiter von der THM haben bei einer Demo mitgemacht. Sie haben gegen Kürzungen von Mitteln für Hochschulen in Hessen demonstriert.

2. Ungefähr 500 Studenten und Mitarbeiter von der THM haben an der Demonstration teilgenommen.

3. Rund 500 Studenten und Mitarbeiter haben bei der Demo mitgemacht.

4. 500 Studenten und Mitarbeiter von der THM haben bei einer Demonstration mitgemacht. Sie haben gegen Kürzungen von Mitteln für Hochschulen in Hessen demonstriert.

5. 500 Studierende und Mitarbeitende von der THM haben bei einer Demonstration mitgemacht.

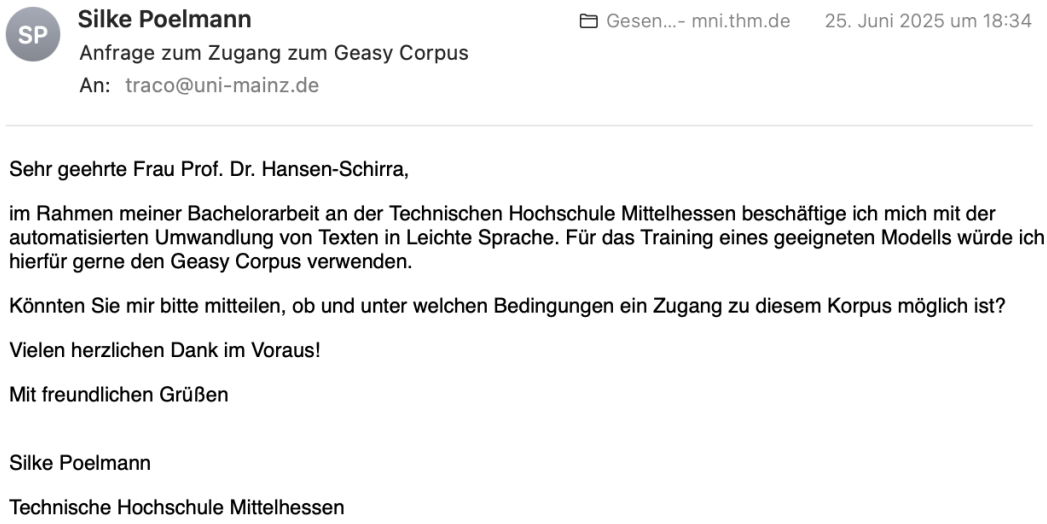
6. B

7. Keine Ausgabe sichtbar

Abbildung A.9: Bewertung des vereinfachten Textes von Leo 13B mit dem Pepper Lesbarkeits-Analyzer.

Abbildung A.10: Bewertung des vereinfachten Textes von Leo 13B mit dem Optimeil Leichte Sprache Prüfer.

A.3 Fine-Tuning



SP **Silke Poelmann** Gesen...- mni.thm.de 25. Juni 2025 um 18:34
Anfrage zum Zugang zum Geasy Corpus
An: traco@uni-mainz.de

Sehr geehrte Frau Prof. Dr. Hansen-Schirra,

im Rahmen meiner Bachelorarbeit an der Technischen Hochschule Mittelhessen beschäftige ich mich mit der automatisierten Umwandlung von Texten in Leichte Sprache. Für das Training eines geeigneten Modells würde ich hierfür gerne den Geasy Corpus verwenden.

Könnten Sie mir bitte mitteilen, ob und unter welchen Bedingungen ein Zugang zu diesem Korpus möglich ist?

Vielen herzlichen Dank im Voraus!

Mit freundlichen Grüßen

Silke Poelmann

Technische Hochschule Mittelhessen

Abbildung A.11: E-Mail-Anfrage zur Nutzung von Geasy Corpus.

Toborek-Korpus

Schritt 1: Repository laden und Umgebung vorbereiten Zunächst wurde das Repository in die Google-Colab-Umgebung geladen, in das Projektverzeichnis gewechselt und alle erforderlichen Programmbibliotheken installiert:

Listing A.1: Klonen und Einrichten des Repositories

```
git clone https://github.com/buschmo/Simple-German-Corpus.git
cd Simple-German-Corpus
pip install -r requirements.txt
python -m spacy download de_core_news_lg
```

Schritt 2: Crawling, Matching und Evaluation

Anschließend wurden die Webinhalte mithilfe des bereitgestellten Crawling-Skripts automatisiert heruntergeladen. Im nächsten Schritt kam das Matching-Skript zum Einsatz, um potenzielle Satzpaare zwischen Standardsprache / Alltagssprache (AS) und Leichter Sprache (LS) zu identifizieren:

Listing A.2: Durchführung von Crawling und Matching

```
python main_crawler.py
python main_matching.py
```

Da das Matching in Google Colab mehrfach abbrach, wurde der Vorgang lokal erneut durchgeführt. Die Berechnung dauerte rund 1,5 Tage. Das Tool bewertete potenzielle Satzpaare hinsichtlich ihrer inhaltlichen Übereinstimmung anhand von Richtigkeit (Precision) und Vollständigkeit (Recall).

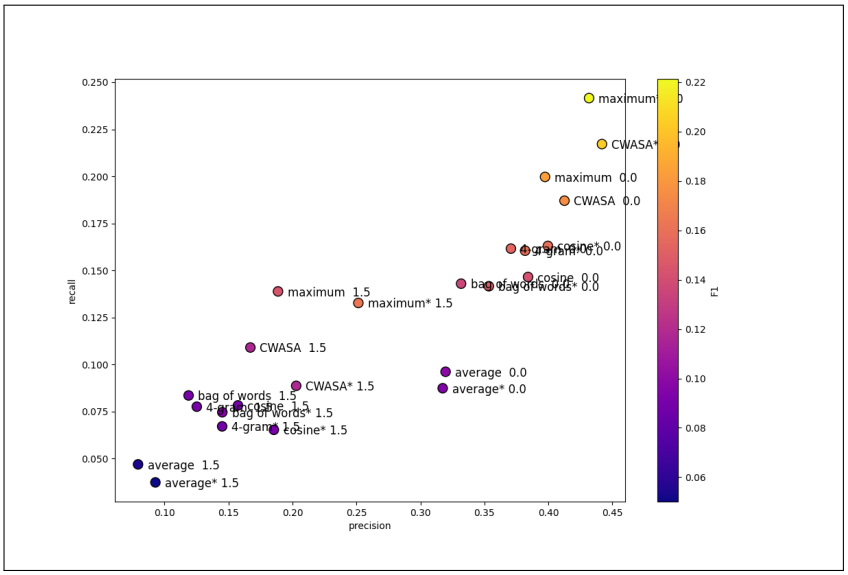


Abbildung A.12: Bewertung der Satzpaar-Erkennung mittels Precision und Recall.

Trotzdem konnten nur **1.321** qualitativ hochwertige Satzpaare identifiziert werden – eine Menge, die für den weiteren Einsatz nicht ausreichte.

```
- Verteilung im Toborek-Korpus -

ERFOLGREICH!
Datei gespeichert: sentence_pairs.csv
Anzahl Satz-Paare: 1,321
Verarbeitete Artikel: 39
```

Zur Erweiterung des Datensatzes wurden daher die Artikelpaare mithilfe experimenteller Python-Skripte manuell zusammengeführt. Die Zuordnung von Standardsprache und Leichter Sprache (LS) erfolgte anhand übereinstimmender Dateinamen, zum Beispiel durch die Kürzel `AS` bzw. `LS` im Dateinamen. Artikel der „taz“ wurden ausgeschlossen, da die Dateinamen keine eindeutige Zuordnung ermöglichten. Auch Artikel der Apotheken-Umschau wurden entfernt, da sie nur in einfacher, nicht in Leichter Sprache vorlagen – um eine klare sprachliche Abgrenzung zu gewährleisten. Abschließend wurden alle CSV-Dateien manuell auf grobe Fehler überprüft.

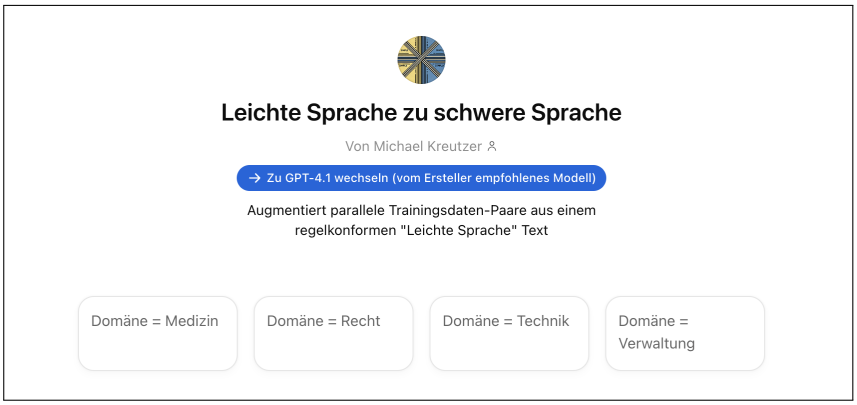


Abbildung A.13: Verteilung der generierten AS-Sätze im synthetischen Korpus (Custom GPT).

Vorbereitung der Trainingsdaten

Schritt 1 und 2: CSV laden, vereinheitlichen und bereinigen

```python
# 1. und 2. Alle CSV-Dateien mit Satzpaaren laden | Spalte vereinheitlichen

import pandas as pd
import glob
import re
import os

# Verzeichnis mit CSV-Dateien (inkl. Unterordner)
path = "/content/drive/MyDrive/Datensaetze/"
all_files = glob.glob(os.path.join(path, "**/*.csv"), recursive=True)

# CSV-Dateien laden und zusammenführen
df = pd.concat([pd.read_csv(f) for f in all_files], ignore_index=True)

# Nur die relevanten Spalten übernehmen und umbenennen
df = df[["AS", "LS"]].dropna()
df.columns = ["input", "target"]  # input = Originaltext, target = Leichte Sprache

# Erfolgsmeldung
print(f" Erfolgreich {len(df)} Datenpaare aus {len(all_files)} Dateien geladen und vereinheitlicht.")
```

Abbildung A.14: Python-Code zum Laden und Vereinheitlichen der CSV-Daten

Schritt 3: Bereinigung der Textdaten

```python
# 3. Clean-Funktion anwenden

def clean(text):
    text = str(text)
    text = re.sub(r"\s+", " ", text.strip())   # Mehrfache Leerzeichen entfernen
    text = re.sub(r"<[^>]+>", "", text)        # HTML-Tags entfernen
    return text

# Texte säubern
df["input"] = df["input"].apply(clean)
df["target"] = df["target"].apply(clean)

# Erfolgsmeldung
print(" Texte in den Spalten 'input' und 'target' wurden erfolgreich bereinigt.")
```

```
Texte in den Spalten 'input' und 'target' wurden erfolgreich bereinigt.
```

Abbildung A.15: Python-Funktion zur Textbereinigung

Schritt 4.1: Erstellung des Testsets für FLAN-T5

```
# 4. Testset für FLAN-T5 erstellen (für Sari-Berechnung)

# Strategischen Prompt hinzufügen
df["input_prompted"] = "Answer the following instruction: Schreibe den Text in leichter deutscher Sprache.\n\nText: " + df["input"] + "\n\nAnswer:"

# Feste Testdaten auswählen
feste_indices = [
    3, 5, 9, 12, 15, 18, 21, 27, 31, 36,
    42, 47, 52, 59, 63, 68, 74, 78, 83, 88,
    94, 99, 105, 111, 118, 123, 129, 134, 140, 147,
    153, 160, 166, 171, 177, 183, 189, 194, 201, 207,
    213, 219, 223, 228, 232, 237, 240, 243, 245, 246
]

# Entsprechende Zeilen extrahieren
test_df = df.iloc[feste_indices]

# Testdaten speichern für Flan-T5
test_df.to_csv("/content/drive/MyDrive/Datensaetze/flan_testset.csv", index=False)

print(f"{len(test_df)} Flan-T5 Testbeispiele gespeichert.")
```

```
50 Flan-T5 Testbeispiele gespeichert.
```

Abbildung A.16: Promptstruktur für FLAN-T5

Schritt 4.2: Erstellung des Testsets für mT5

```
# 4. Testset für mT5 erstellen (für Sari-Berechnung)

# Strategischen Prompt hinzufügen
df["input_prompted"] = "Schreibe diesen Text in leichter deutscher Sprache: " + df["input"]"

# Gleiche feste Testdaten auswählen (für Vergleichbarkeit)
feste_indices = [
    3, 5, 9, 12, 15, 18, 21, 27, 31, 36,
    42, 47, 52, 59, 63, 68, 74, 78, 83, 88,
    94, 99, 105, 111, 118, 123, 129, 134, 140, 147,
    153, 160, 166, 171, 177, 183, 189, 194, 201, 207,
    213, 219, 223, 228, 232, 237, 240, 243, 245, 246
]

# Entsprechende Zeilen extrahieren
test_df = df.iloc[feste_indices]

# Testdaten speichern für mT5
test_df.to_csv("/content/drive/MyDrive/Datensaetze/mt5_testset.csv", index=False)

print(f"{len(test_df)} mT5 Testbeispiele gespeichert.")
```

```
50 mT5 Testbeispiele gespeichert.
```

Abbildung A.17: Promptstruktur für mT5

Schritt 5.1: Promptgenerierung für FLAN-T5

```
import random
import pandas as pd

# 5. Prompt-Generierung für's Training (Baseline vs. Strategien) FLAN-T5

# Paper-konforme Prompt-Strategien für Flan-T5
ba_prompts = [
    "Answer the following instruction: Schreibe den Text in leichter deutscher Sprache.\n\nText: {}\n\nAnswer:",
    "Answer the following instruction: Vereinfache den Text für Menschen mit Lernschwierigkeiten. Verwende einfache Wörter und kurze Sätze.\n\nText: {}\n\nAnswer:",
    "Answer the following instruction: Vereinfache den Text für Menschen mit Lernschwierigkeiten. Verwende keine Fremdwörter und keine langen Sätze.\n\nText: {}\n\nAnswer:"
]

weights = [40, 35, 25]  # Wie im ursprünglichen Code

# Trainingsdaten erzeugen
training_data = []
for _, row in df.iterrows():
    prompt = random.choices(ba_prompts, weights=weights, k=1)[0]
    training_data.append({
        "input": prompt.format(row["input"]),
        "target": row["target"]
    })

training_df = pd.DataFrame(training_data)

# Baseline separat erzeugen (paper-konform)
baseline_df = df.copy()
baseline_df["input"] = "Answer the following instruction: Schreibe den Text in leichter deutscher Sprache.\n\nText: " + baseline_df["input"] + "\n\nAnswer:"

# Speichern
baseline_df.to_csv("/content/drive/MyDrive/Datensaetze/flan_gesamt_mit_prompt.csv", index=False)
training_df.to_csv("/content/drive/MyDrive/Datensaetze/flan_training_strategies.csv", index=False)

print(f"{len(training_df)} Strategische Trainingsbeispiele gespeichert.")
print(f"{len(baseline_df)} Baseline Trainingsbeispiele gespeichert.")
```

```
265 Strategische Trainingsbeispiele gespeichert.
265 Baseline Trainingsbeispiele gespeichert.
```

Abbildung A.18: Promptgenerierung mit Strategien und Baseline für FLAN-T5

Schritt 5.2: Promptgenerierung für mT5

```python
import random
import pandas as pd

# 5. Prompt-Generierung für's Training (Baseline vs. Strategien) mT5

# Completion-style Prompt-Strategien für mT5
ba_prompts = [
    "Schreibe diesen Text in leichter deutscher Sprache: {}",
    "Vereinfache den Text für Menschen mit Lernschwierigkeiten. Verwende einfache Wörter und kurze Sätze: {}",
    "Vereinfache den Text für Menschen mit Lernschwierigkeiten. Verwende keine Fremdwörter und keine langen Sätze: {}"
]

weights = [40, 35, 25]  # Wie im ursprünglichen Code

# Trainingsdaten erzeugen
training_data = []
for _, row in df.iterrows():
    prompt = random.choices(ba_prompts, weights=weights, k=1)[0]
    training_data.append({
        "input": prompt.format(row["input"]),
        "target": row["target"]
    })

training_df = pd.DataFrame(training_data)

# Baseline separat erzeugen (completion-style)
baseline_df = df.copy()
baseline_df["input"] = "Text: " + baseline_df["input"] + "\nLeichte Sprache:"

# Speichern
baseline_df.to_csv("/content/drive/MyDrive/Datensaetze/mt5_gesamt_mit_prompt.csv", index=False)
training_df.to_csv("/content/drive/MyDrive/Datensaetze/mt5_training_strategies.csv", index=False)

print(f"{len(training_df)} mT5 Strategische Trainingsbeispiele gespeichert.")
print(f"{len(baseline_df)} mT5 Baseline Trainingsbeispiele gespeichert.")
```

```
265 mT5 Strategische Trainingsbeispiele gespeichert.
265 mT5 Baseline Trainingsbeispiele gespeichert.
```

Abbildung A.19: Promptgenerierung mit Strategien und Baseline für mT5

```python
# SARI-Bewertung vor dem Fine-Tuning

# Pakete importieren
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
import pandas as pd
import torch
from tqdm import tqdm
import evaluate

# mT5-Base Modell laden
model_name = "google/mT5-large" #"google/flan-t5-large"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
model.eval()
model.to("cuda")

# Testdaten laden
# → CSV muss die Spalten enthalten: input, target, input_prompted
test_df = pd.read_csv("/content/drive/MyDrive/Datensaetze/mt5_testset.csv")
sources = test_df["input"].tolist()
references = [[t] for t in test_df["target"].tolist()]
inputs_for_model = test_df["input_prompted"].tolist()

# Vorhersagen generieren
predictions = []
for input_text in tqdm(inputs_for_model, desc="Generiere Vorhersagen"):
    encoded = tokenizer(
        input_text,
        return_tensors="pt",
        truncation=True,
        padding=True,
        max_length=768
    ).to("cuda")

    with torch.no_grad():
        output_ids = model.generate(
            encoded["input_ids"],
            max_new_tokens=512
        )
        output_text = tokenizer.decode(output_ids[0], skip_special_tokens=True)
        predictions.append(output_text)

# SARI berechnen
sari = evaluate.load("sari")
results = sari.compute(
    predictions=predictions,
    references=references,
    sources=sources
)

# Ergebnis anzeigen
print("\n" + "="*60)
print("  SARI SCORE MIT GOOGLE/MT5-LARGE")
print("="*60)
print(f"SARI-Score: {results['sari']:.2f}")
```

Abbildung A.20: Quellcode zur Sari-Berechnung von Flan-T5 und mT5.

A.4 Implementierung

Abbildung A.21: Mobile Ansicht: Archivübersicht aller gespeicherter Texte

Abbildung A.22: Mobile Ansicht: Detailansicht eines gespeicherten Textes

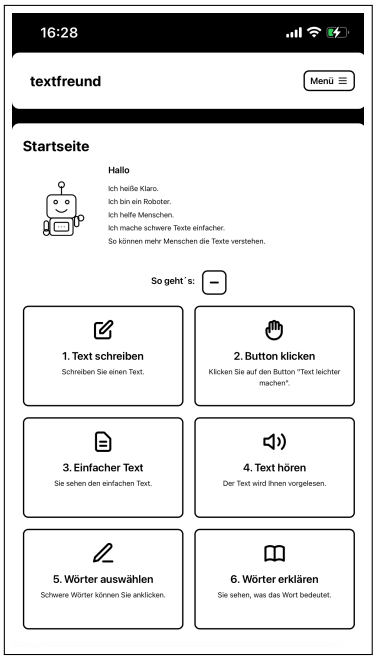


Abbildung A.23: Mobile Ansicht: Startseite mit Klaro und Hilfekacheln

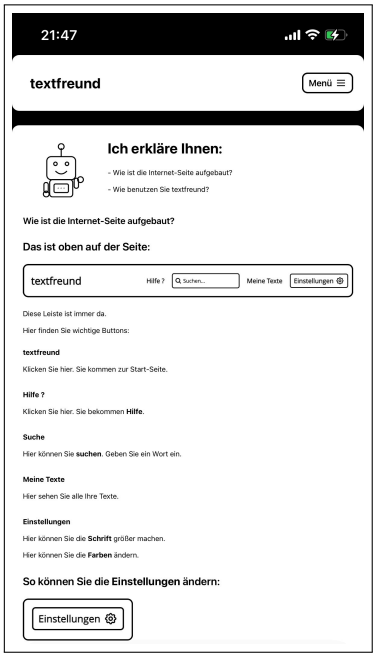


Abbildung A.24: Mobile Ansicht: Statische Seite: Hilfe

Abbildung A.25: Mobile Ansicht: Statische Seite: Kontakt

Abbildung A.26: Mobile Ansicht: Statische Seite: Impressum

Abbildung A.27: Mobile Ansicht: Statische Seite: Datenschutz