▲ Hochschule Harz

Hochschule für angewandte Wissenschaften

Methodik eines Ingenieurs

Vorgelegt von:

Philipp Thüring
m26665
Friedrichstraße 12
38855 Wernigerode

Erstprüfer	Johr	Alexander
Zweitprüfer	Singer	Jürgen
Abgabedatum	28. Februar	2021

Inhaltsverzeichnis

Hauptkapitel der Masterarbeit (neue Gliederung)

1 Einleitung

Die Digitalisierung des öffentlichen Personennahverkehrs erfordert zunehmend die automatisierte Verarbeitung und Aufbereitung von Informationen für unterschiedliche Zielgruppen und Kommunikationskanäle. Verkehrsbetriebe stehen dabei vor der Herausforderung, technische Betriebsinformationen in verständliche Fahrgastinformationen zu transformieren. Diese Transformation erfolgt gegenwärtig überwiegend manuell und ist sowohl zeitintensiv als auch anfällig für Inkonsistenzen. Mit der zunehmenden Verfügbarkeit großer Sprachmodelle (Large Language Models, LLM) eröffnen sich neue Möglichkeiten zur Automatisierung solcher Transformationsprozesse. Gleichzeitig führt der Einsatz dieser Modelle häufig zu einer Überkapazität an Rechenressourcen, wenn generische Großmodelle für spezialisierte Aufgaben eingesetzt werden, die auch von kompakteren Systemen bewältigt werden könnten.

Die vorliegende Arbeit untersucht, wie ein ressourceneffizientes System zur automatischen Transformation von Verkehrsanweisungen entwickelt werden kann, indem ein kompaktes Sprachmodell durch gezieltes Fine-Tuning auf die spezifische Aufgabe spezialisiert wird. Dabei steht die Frage im Mittelpunkt, welche Qualität durch ein 7-Milliarden-Parameter-Modell mit begrenzten Trainingsdaten erreicht werden kann und wie verschiedene Quantisierungstechniken die Ressourceneffizienz bei gleichbleibender oder verbesserter Qualität beeinflussen.

1.1 Motivation und Problemstellung

Die Leipziger Verkehrsbetriebe (LVB) erstellen für betriebliche Störungen und geplante Baumaßnahmen technische Verkehrsanweisungen, die detaillierte Informationen zu Umleitungen, Haltestellenverlegungen und Fahrplanänderungen enthalten. Diese Anweisungen werden intern von Mitarbeitenden verfasst und richten sich primär an das Fachpersonal. Für die Fahrgastkommunikation müssen diese technischen Dokumente jedoch in verschiedene Formate transformiert werden, die sich in Detailgrad, Formulierung und Struktur unterscheiden. Dieser Transformationsprozess erfolgt gegenwärtig manuell durch Mitarbeitende, die die komplexen betrieblichen Informationen in zielgruppengerechte Formulierungen übersetzen.

Die manuelle Transformation stellt mehrere Herausforderungen dar. Erstens ist der Prozess zeitintensiv, da für jede Verkehrsanweisung mehrere unterschiedliche Textversionen erstellt werden müssen. Zweitens können durch die manuelle Bearbeitung Inkonsistenzen in Formulierung und Struktur entstehen, insbesondere wenn verschiedene Mitarbeitende an ähnlichen Transformationen arbeiten. Drittens wächst mit zunehmender Anzahl von Baumaßnahmen und Störungen der Aufwand für die Erstellung und Pflege der Fahrgastinformationen.

Die Automatisierung dieses Prozesses mittels künstlicher Intelligenz erscheint

naheliegend. In der Praxis zeigt sich jedoch eine grundlegende Problematik moderner KI-Implementierungen: Häufig werden überdimensionierte Modelle für Aufgaben eingesetzt, die auch von kleineren, spezialisierten Systemen bewältigt werden könnten. Diese Tendenz zur Überkapazität führt zu unnötig hohen Betriebskosten, erhöhtem Energieverbrauch und erschwert das Deployment auf Consumer-Hardware. Generische Großmodelle mit mehreren hundert Milliarden Parametern benötigen spezialisierte Server-Infrastruktur und verursachen laufende Kosten durch Cloud-APIs oder erfordern den Betrieb eigener Hochleistungsserver.

Für die LVB bietet sich daher die Möglichkeit, von Beginn an auf eine nachhaltige und ressourceneffiziente Lösung zu setzen. Ein gezielt trainiertes, kompaktes Modell kann die spezifische Aufgabe der Texttransformation effizient erfüllen, ohne die Nachteile generischer Großmodelle in Kauf nehmen zu müssen. Dies ermöglicht potentiell den Betrieb auf Consumer-Hardware im Unternehmen, reduziert laufende Kosten und gewährleistet gleichzeitig Datenschutz durch lokale Verarbeitung.

1.2 Zielsetzung und Forschungsfragen

Das Hauptziel dieser Arbeit besteht in der Entwicklung eines ressourceneffizienten Systems zur automatischen Transformation technischer Verkehrsanweisungen in einheitlich formulierte Fahrgastinformationen. Dabei soll bewusst ein kleineres Sprachmodell durch gezieltes Fine-Tuning auf die domänenspezifische Aufgabe spezialisiert werden, um langfristig Rechenressourcen zu schonen und eine nachhaltige Lösung zu etablieren.

Die Arbeit orientiert sich an folgenden Forschungsfragen:

1. Wie kann ein 7-Milliarden-Parameter-Modell durch domänenspezifisches Fine-Tuning für die Transformation von Verkehrsanweisungen optimiert werden, und welche Qualität lässt sich mit begrenzten Trainingsdaten erreichen?
2. Welche Auswirkungen haben unterschiedliche Quantisierungsstufen (Vollmodell, 8-Bit, 4-Bit) auf die Qualität der Texttransformation, und existieren überraschende Befunde hinsichtlich des Trade-offs zwischen Ressourceneffizienz und Ausgabequalität?
3. Wie kann die Transformation durch systematisches Prompt Engineering automatisiert werden, um konsistente und regelkonforme Ausgaben zu gewährleisten?

Die erste Forschungsfrage adressiert die grundlegende Machbarkeit der Aufgabe mit einem kompakten Modell. Im Gegensatz zu generischen Großmodellen mit mehreren hundert Milliarden Parametern soll untersucht werden,

ob ein 7B-Modell durch spezialisiertes Training ausreichende Qualität für die domänenspezifische Transformation erreichen kann. Dabei ist insbesondere relevant, wie sich die begrenzte Verfügbarkeit von Trainingsdaten auswirkt, da für die Leipziger Verkehrsbetriebe keine umfangreichen vorannotierten Datensätze existieren.

Die zweite Forschungsfrage fokussiert auf die praktische Deploybarkeit des Systems. Quantisierung reduziert den Speicherbedarf und Rechenaufwand von Sprachmodellen erheblich, kann jedoch mit Qualitätseinbußen einhergehen. Die systematische Evaluation verschiedener Quantisierungsstufen soll aufzeigen, welcher Trade-off zwischen Ressourceneffizienz und Ausgabequalität für den praktischen Einsatz optimal ist.

Die dritte Forschungsfrage betrifft die Integration domänenspezifischen Wissens in das System. Durch gezieltes Prompt Engineering können Informationen zu Linienbezeichnungen, Fahrzeugtypen und Formatierungskonventionen in die Generierung integriert werden, ohne auf komplexe Retrieval-Augmented Generation Systeme zurückgreifen zu müssen. Dies reduziert die Systemkomplexität und ermöglicht eine effizientere Verarbeitung.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in neun Kapitel, die aufeinander aufbauen und die systematische Entwicklung sowie Evaluation des ressourceneffizienten Transformationssystems dokumentieren.

Kapitel 2 legt die theoretischen Grundlagen der Arbeit dar. Es werden die Fundamente des Natural Language Processing eingeführt, beginnend mit der Transformer-Architektur über das Pretraining-Finetuning-Paradigma bis hin zu deutschsprachigen Modellen. Anschließend werden Methoden zur Modelloptimierung behandelt, insbesondere parameter-effizientes Fine-Tuning durch LoRA, Strategien für das Training mit begrenzten Daten sowie Prompt Engineering. Der dritte Teil fokussiert auf Aspekte der Produktionsreife, einschließlich Quantisierung, Halluzination Prevention und Knowledge Integration.

Kapitel 3 beschreibt den Anwendungskontext der Leipziger Verkehrsbetriebe. Es charakterisiert die Struktur technischer Verkehrsanweisungen anhand von sieben identifizierten Komplexitätsmerkmalen und dokumentiert die Extraktion von Transformationsregeln aus bestehenden Fahrgastinformationen. Die exemplarische Analyse konkreter Verkehrsanweisungen verdeutlicht die Herausforderungen der Transformationsaufgabe.

Kapitel 4 begründet die Auswahl des Modells leo-mistral-hessianai-7b. Es werden drei Kandidaten verglichen: Mistral-7B als englisches Base Model, Mistral-7B-Instruct als englisches Instruct-Model sowie LeoLM als Base Model mit deutschem Continued Pretraining. Die Entscheidung für LeoLM wird durch Anforde-

rungen an deutschsprachige Kompetenz, Deploybarkeit auf Consumer-Hardware und Eignung für domänenspezifisches Fine-Tuning begründet.

Kapitel 5 dokumentiert die Entwicklung des Datensatzes. Es beschreibt die Annotationsstrategie für die manuelle Erstellung der Trainingsbeispiele, das implementierte Qualitätssicherungssystem, sieben entwickelte Augmentierungsstrategien zur Erweiterung des begrenzten Datensatzes sowie die finale Datensatzstruktur mit Train-Validation-Test-Split. Das Kapitel schließt mit der Diskussion der Herausforderungen beim Training mit 950 Beispielen.

Kapitel 6 beschreibt die technische Implementierung des Systems. Es dokumentiert die Entwicklungsumgebung und eingesetzte Hardware, das Fine-Tuning mittels QLoRA auf einer NVIDIA RTX A2000 12GB Grafikkarte sowie das Design der Prompt-Templates für die automatisierte Transformation. Die Integration domänenspezifischen Wissens durch Knowledge-Enhanced System Prompts wird detailliert dargelegt, ebenso die Implementierung automatisierter Qualitätssicherung zur Erkennung von Halluzinationen und semantischen Inkonsistenzen.

Kapitel 7 präsentiert die Evaluation des Systems. Es beschreibt die Evaluationsmethodik einschließlich des Testdatensatzes mit 38 Beispielen sowie die eingesetzten automatischen und manuellen Evaluationsmetriken.

Kapitel 8 diskutiert die Ergebnisse der Arbeit. Es ordnet den Quantisierungsbefund theoretisch ein, reflektiert Limitationen des Systems und bewertet die praktische Anwendbarkeit im Unternehmenskontext der Leipziger Verkehrsbetriebe.

Kapitel 9 fasst die Erkenntnisse zusammen und gibt einen Ausblick auf zukünftige Entwicklungen. Es werden Erweiterungsmöglichkeiten des Systems diskutiert, insbesondere die Integration in umfassendere Automatisierungsszenarien und die potenzielle Evaluation größerer Modellvarianten. Die Übertragbarkeit des Ansatzes auf andere Verkehrsbetriebe und verwandte Domänen wird abschließend erörtert.

2 Theoretische Grundlagen und Stand der Forschung

Die in Kapitel 1 dargelegte Problemstellung der automatisierten Transformation von LVB-Verkehrsanweisungen erfordert fundierte Expertise im Bereich Natural Language Processing. Diese Transformation stellt keine triviale Übersetzungsaufgabe dar, sondern verlangt die simultane Berücksichtigung von Kontextabhängigkeit, fachlicher Präzision und allgemeiner Verständlichkeit bei der Konversion von Fachsprache in Allgemeinsprache [1].

Das vorliegende Kapitel strukturiert die theoretischen Grundlagen entlang dreier komplementärer Dimensionen. Abschnitt 2.1 etabliert die NLP-Fundamente, beginnend mit der Transformer-Architektur als paradigmatischem Durchbruch in der Sprachverarbeitung, über das Pretraining-Finetuning-Paradigma bis hin zu deutschsprachigen Modellen, insbesondere der Entwicklung von Mistral zu LeoLM. Abschnitt 2.2 fokussiert auf Anpassungsmethodik, wobei Parameter-effizientes Fine-Tuning durch Low-Rank Adaptation sowie Datenstrategien bei limitierten Ressourcen behandelt werden. Abschnitt 2.3 adressiert Aspekte der Produktionsreife, einschließlich Quantisierung für Deployment unter Hardware-Constraints sowie Halluzination Prevention zur Sicherstellung von Faktentreue.

Der rote Faden verbindet theoretische Konzepte mit praktischen Constraints: Kapitel 3 demonstriert, wie der Anwendungskontext technologische Entscheidungen determiniert, während Kapitel 6 die Operationalisierung theoretischer Konzepte in der Implementierung darlegt. Das vorliegende Kapitel fundiert die entwickelte Lösung in etablierter Forschung und schafft die konzeptionelle Basis für die nachfolgenden Kapitel [2].

2.1 Natural Language Processing für Verkehrsinformationen

Die Anwendung von Natural Language Processing im Verkehrssektor bewegt sich im Spannungsfeld zwischen universellen Sprachmodellen und domänenspezifischen Anforderungen. Während proprietäre Systeme wie GPT-4 oder Claude beeindruckende Generalfähigkeiten demonstrieren, erfordert die Verkehrsdomäne spezifische Terminologie, rechtliche Präzision und Lokalkontexte, die eine spezialisierte Adaption notwendig machen [3]. Hinzu tritt die Notwendigkeit von Open-Source-Lösungen, motiviert durch Datenschutz-Grundverordnung-Konformität, Datensouveränität und Kosteneffizienz [4].

Die Transformation von Verkehrsinformationen erfordert simultanes Kontextverständnis, Stiladaption und Faktentreue. Der qualitative Unterschied zwischen einer knappen Feststellung wie „Linie 10 fährt nicht“ und einer vollständigen Information „Straßenbahn der Linie 10 verkehrt aufgrund von Bauarbeiten...“

illustriert die Notwendigkeit impliziten Wissens über Linientypen, Fahrzeugkategorien und Alternativrouten. Moderne Large Language Models auf Basis der Transformer-Architektur vereinen Flexibilität und Generalisierung, bringen jedoch neue Herausforderungen bezüglich Kontrolle und Faktentreue mit sich [5].

Die nachfolgenden Unterabschnitte strukturieren diese Grundlagen: Abschnitt 2.1.1 behandelt technische Fundamente, Abschnitt 2.1.2 diskutiert verkehrsspezifische NLP-Anwendungen im Stand der Technik, und Abschnitt 2.1.3 fokussiert auf deutschsprachige und domänenspezifische Modelle, insbesondere die LeoLM-Familie.

2.1.1 Grundlagen der Sprachverarbeitung

Die automatisierte Verarbeitung und Transformation von Verkehrsinformationen stellt hohe Anforderungen an Natural Language Processing-Systeme. Moderne Ansätze der Sprachverarbeitung basieren auf der Transformer-Architektur, die seit ihrer Einführung im Jahr 2017 die Entwicklung von Sprachmodellen maßgeblich geprägt hat [6]. Das in dieser Arbeit verwendete Modell LeoLM-7B baut auf dieser Architektur auf und nutzt deren Vorteile für die Verarbeitung deutschsprachiger Texte.

Im Gegensatz zu früheren sequenziellen Ansätzen basiert die Transformer-Architektur auf dem Self-Attention-Mechanismus, der es ermöglicht, kontextabhängige Repräsentationen für jedes Token zu berechnen. Durch Multi-Head Attention können unterschiedliche Aspekte der Kontextbeziehungen erfasst werden [6]. Diese Architektur ermöglicht die parallele Verarbeitung von Sequenzen und führt zu deutlich schnelleren Trainingszeiten sowie besserer Skalierbarkeit.

Für die in dieser Arbeit behandelte Aufgabe der Textgenerierung und -transformation ist die Decoder-Only-Architektur besonders geeignet, da sie speziell für die autoregressive Erzeugung von Text konzipiert wurde [7]. Diese Architektur bildet die Grundlage des verwendeten LeoLM-Modells.

Ein zentrales Konzept moderner NLP-Systeme ist das Pretraining auf großen, unlabeled Textkorpora. Durch Self-Supervised Learning entwickeln diese Modelle ein umfassendes Sprachverständnis inklusive Syntax, Semantik und implizitem Weltwissen [8].

Für die vorliegende Arbeit ist insbesondere die Mistral-7B-Architektur von Bedeutung, da sie die Grundlage für das verwendete LeoLM-Modell bildet. Mistral 7B ist ein hocheffizienter Decoder-Only-Transformer mit 7 Milliarden Parametern, der mehrere innovative Architekturmerkmale aufweist [9]. Grouped-Query Attention reduziert die Größe des Key-Value-Cache und ermöglicht schnellere Inferenz. Sliding Window Attention erlaubt die effiziente Verarbeitung langer Kontexte, während der Rolling Buffer Cache die Speichernutzung optimiert.

Mit 7 Milliarden Parametern stellt Mistral einen Sweet Spot zwischen Modellleistung und Ressourceneffizienz dar und übertrifft in Benchmarks viele deutlich größere Modelle [9].

Ein wichtiger Unterschied besteht zwischen Base Models und Instruct Models. Base Models wie Mistral-7B sind auf reines Language Modeling trainiert und setzen primär Texte fort, ohne notwendigerweise expliziten Anweisungen zu folgen. Instruct Models hingegen durchlaufen zusätzlich ein Instruction Tuning, bei dem sie mittels Supervised Fine-Tuning auf Instruktionsdatensätzen trainiert werden, um gezielt Anweisungen zu befolgen [10]. Optional kann dieser Prozess durch Reinforcement Learning from Human Feedback weiter verfeinert werden.

Für deutschsprachige Anwendungen ist die LeoLM-Familie von besonderer Relevanz. Diese Modelle nutzen Mistral-7B als Basis und durchlaufen ein Continued Pretraining auf deutschen Textkorpora [11]. Das in dieser Arbeit verwendete Modell leo-mistral-hessianai-7b ist eine Base-Variante, die speziell für die deutsche Sprache optimiert wurde. Diese Adaption kombiniert die architektonischen Vorteile und die Effizienz von Mistral mit erhöhter Sprachkompetenz im Deutschen.

Die Tokenization erfolgt bei Mistral mittels Byte Pair Encoding mit einem Vokabular von 32.000 Tokens, wobei LeoLM einen an die deutsche Morphologie angepassten Tokenizer verwendet, der besser mit Komposita, Umlauten und anderen sprachspezifischen Besonderheiten umgehen kann [12].

Das Konzept des Transfer Learning bildet die theoretische Grundlage für die Nutzung vortrainierter Modelle in spezifischen Anwendungsdomänen. Transfer Learning bezeichnet den Wissenstransfer von einer Source Domain, in der das Modell vortrainiert wurde, zu einer Target Domain, für die es angepasst werden soll [13]. Dieser Ansatz reduziert den Bedarf an aufgabenspezifischen Trainingsdaten und die erforderliche Trainingszeit erheblich.

Das etablierte Pretrain-Finetune-Paradigma verläuft in zwei Phasen: Zunächst erfolgt das Pretraining auf großen, unlabeled Textkorpora mittels unsupervised Learning, wodurch das Modell grundlegendes Sprachverständnis erwirbt. In der zweiten Phase wird das Modell mittels Fine-Tuning auf eine spezifische Aufgabe angepasst, wobei supervised Learning mit aufgabenspezifischen Daten zum Einsatz kommt [14]. Im Kontext dieser Arbeit bedeutet dies die Spezialisierung auf die Transformation von LVB-Verkehrsanweisungen.

Die Vorteile von Transfer Learning sind vielfältig: Vortrainierte Gewichte dienen als optimaler Startpunkt und beschleunigen das Training erheblich. Das bereits vorhandene Sprachverständnis führt zu besserer Generalisierung auf neue Daten [13]. Dennoch bestehen Herausforderungen: Catastrophic Forgetting bezeichnet den Verlust vortrainierter Fähigkeiten während des Fine-Tunings. Der Domain Shift zwischen Pretraining-Daten und Zieldomäne kann zu Leistungseinbußen führen, insbesondere wenn sich Vokabular oder Sprachstil deutlich unterscheiden

[15]. Bei kleinen Datensätzen besteht zudem die Gefahr des Overfittings.

Für die vorliegende Arbeit ist besonders relevant, dass LeoLM bereits auf umfangreichen deutschen Textkorpora vortrainiert wurde, was eine solide Basis für die weitere Spezialisierung bildet. Das Fine-Tuning auf den vergleichsweise kleinen Datensatz der LVB-Verkehrsanweisungen wird durch Transfer Learning erst praktikabel und ermöglicht die notwendige domänenspezifische Anpassung an die fachsprachlichen Anforderungen des Verkehrssektors.

2.1.2 NLP-Anwendungen im Verkehrssektor

Natural Language Processing fungiert als Enabler-Technologie für moderne Verkehrsmanagement-Strategien. Die technologische Evolution vollzog sich vom regelbasierten Mustererkennen zu bedeutungsverstehenden Large Language Models, wodurch die systematische Automatisierung der Informationsverarbeitung ermöglicht wurde [16]. Die Anwendung von Deep Learning-Methoden im intelligenten Verkehrswesen umfasst dabei ein breites Spektrum von Aufgaben, darunter Verkehrsprognose, Unfallanalyse und die Verarbeitung natürlichsprachlicher Anfragen.

Die Kernrelevanz für die vorliegende Untersuchung manifestiert sich in der automatisierten Berichtserstellung und Dokumentenanalyse. Empirische Studien demonstrieren substantielle Fortschritte bei der Zusammenfassung von Unfallberichten und der erfolgreichen Klassifizierung der Unfall-Schwere aus Textbeschreibungen [17]. Zudem belegen Untersuchungen die Effektivität von LLMs bei der Analyse und Aufbereitung von Fahrgastanfragen. Eine systematische Literaturübersicht zu NLP-Anwendungen in intelligenten Verkehrssystemen zeigt die zunehmende Bedeutung sprachverarbeitender Methoden für die Transformation und Aufbereitung verkehrsbezogener Informationen.

Die Transformation technischer Informationen in Endnutzerformate erfordert simultane Berücksichtigung von Fachpräzision und Verständlichkeit. Diese Brückenfunktion zwischen technischer Fachsprache und Allgemeinverständlichkeit konstituiert die zentrale Aufgabe der Verkehrsanweisungs-Transformation und unterscheidet sich fundamental von reiner Klassifizierung durch ihre essenzielle generative Komponente [18].

Die empirischen Erfolge rechtfertigen die Hypothese, dass LLMs für Verkehrsanweisungs-Transformation geeignet sind. Gleichwohl persistieren domänenspezifische Herausforderungen: Spezialisiertes Vokabular erfordert gezielte Adaption, Faktentreue muss bei sicherheitskritischen Informationen absolute Priorität genießen, und Konsistenz stellt deterministische Anforderungen an probabilistische Systeme [19].

2.1.3 Domänenspezifische Sprachmodelle

Die Forschung dokumentiert konsistente Überlegenheit domänenspezifisch angepasster Modelle gegenüber General-Purpose-Systemen in verkehrsspezifischen Aufgaben [20]. Existierende spezialisierte Systeme eignen sich jedoch nicht für deutschsprachige Fahrgastinformation.

Verschiedene spezialisierte Systeme wurden für den Verkehrssektor entwickelt, sind jedoch primär englischsprachig und auf andere Anwendungsfälle fokussiert. Die Systeme konzentrieren sich auf Verkehrssicherheit, chinesischsprachige Wissensvermittlung oder englischsprachige Query-Beantwortung mit Echtzeit-Datenbanken [3]. Multimodale Modelle fokussieren auf autonome Fahrzeuge und Predictive Analytics.

Die systematische Analyse offenbart fünf Hindernisse für die Verwendung existierender Systeme: erstens die Sprachbarriere, da alle Systeme englisch- oder chinesischsprachig sind. Zweitens ein Aufgaben-Mismatch, da Safety, Prediction und Simulation nicht äquivalent zu Text-Transformation sind. Drittens Lizenzierungsprobleme, da meist proprietäre oder Closed-Source-Basis vorliegt. Viertens Infrastrukturanforderungen, da Echtzeit-Datenbanken erforderlich sind. Fünftens Deployment-Beschränkungen, da die Systeme nicht für lokale, offline-fähige Ausführung konzipiert sind.

Daraus erwächst die Notwendigkeit eigenständiger Modellentwicklung mit der Strategie: Deutsches Basismodell plus domänenspezifisches Fine-Tuning. Diese Rechtfertigung adressiert eine Forschungslücke bei deutschsprachiger Verkehrsinformation, deren detaillierte Begründung in Kapitel 4 erfolgt. Die theoretische Überlegenheit domänenspezifischer Modelle etabliert, stellt sich nun die Frage: Wie erstellt man solche Modelle? Abschnitt 2.2 behandelt Fine-Tuning-Methoden, Datenstrategien und Optimierungen [21].

2.2 Sprachmodelle und Fine-Tuning

Die Problemstellung der Modellanpassung manifestiert sich im Spannungsfeld zwischen allgemeinem Sprachverständnis und domänenspezifischer Expertise. Vortrainierte Sprachmodelle verfügen über umfassendes Verständnis natürlicher Sprache, erworben durch Training auf Milliarden von Tokens. Die Spezialisierung auf domänenspezifische Aufgaben erfordert jedoch gezielte Anpassung [14].

Ressourcenbeschränkungen beim Full Fine-Tuning stellen prohibitive Barrieren dar. Die Anpassung sämtlicher Modellparameter erfordert Training auf Multi-GPU-Clustern mit Speicherbedarf über 100 GB VRAM [22]. Die Constraint dieser Arbeit, nämlich Consumer-Hardware mit begrenzten Ressourcen, erzwingt die Anwendung parameter-effizienter Fine-Tuning-Methoden.

Das Risiko des Overfitting bei vollständiger Parameteranpassung mit kleinem

Datensatz ist substanziell. Parameter-effiziente Verfahren bieten den Vorteil reduzierten Datenbedarfs durch gezielte Adaption und wirken dem Overfitting entgegen [23].

Die methodischen Ansätze gliedern sich in vier Kategorien: Parameter-effiziente Fine-Tuning-Verfahren reduzieren die Anzahl trainierbarer Parameter (Abschnitt 2.2.1). Prompt Engineering fungiert als komplementärer Ansatz zur Leistungssteigerung ohne Parametermodifikation (Abschnitt 2.2.2). Datenaufbereitung und Augmentation adressieren limitierte Datenverfügbarkeit (Abschnitt 2.2.3). Gegenmaßnahmen zu Overfitting und Catastrophic Forgetting sichern die Generalisierungsfähigkeit (Abschnitt 2.2.4).

2.2.1 Fine-Tuning-Methoden

Das Training großer vortrainierter Sprachmodelle für spezifische Downstream-Aufgaben stellt eine fundamentale Herausforderung dar. Während das klassische Full Fine-Tuning alle Modellparameter anpasst, führt diese Strategie bei wachsenden Modellgrößen zu zunehmend unpraktikablen Ressourcenanforderungen [22]. Ein Training auf Multi-GPU-Clustern kann mehrere Wochen dauern und Kosten im vier- bis fünfstelligen Bereich verursachen.

Als Antwort auf diese Herausforderung haben sich Parameter-Efficient Fine-Tuning-Methoden etabliert, die eine Modellspezialisierung mit drastisch reduzierten Rechenressourcen ermöglichen. Die PEFT-Landschaft lässt sich in drei Hauptkategorien einteilen [24]: Additive Fine-Tuning fügt zusätzliche trainierbare Module zum gefrorenen Basismodell hinzu. Zu dieser Kategorie zählen Methoden wie Adapter, Prefix Tuning und Prompt Tuning. Reparameterized Fine-Tuning hingegen zerlegt Gewichtsupdates in niedrigrangige Matrizen. Der prominenteste Vertreter ist Low-Rank Adaptation, dessen entscheidender Vorteil darin besteht, dass durch Post-Training Weight Merging keine Inferenz-Latenz entsteht. Die dritte Kategorie, Selective Fine-Tuning, trainiert nur ausgewählte Teilmengen existierender Parameter und findet in dieser Arbeit keine Anwendung.

Low-Rank Adaptation basiert auf der Hypothese, dass Gewichtsänderungen während der Modelladaptation einen niedrigen intrinsischen Rang aufweisen [25]. Das Update wird als Produkt zweier niedrigrangiger Matrizen dargestellt: $\Delta W = BA$, wobei $B \in \mathbb{R}^{d \times r}$ und $A \in \mathbb{R}^{r \times k}$ mit $r \ll d, k$. Die finale Gewichtsmatrix ergibt sich als $W = W_0 + BA$, wobei W_0 die eingefrorenen Originalgewichte darstellt und der Rang r als Hyperparameter die Balance zwischen Expressivität und Effizienz kontrolliert.

Während des Trainings bleiben die vortrainierten Gewichte eingefroren, und ausschließlich die Low-Rank-Matrizen werden trainiert. Die Effizienzgewinne von LoRA sind substanziell: Eine bis zu 10.000-fache Reduktion trainierbarer Parameter bei vergleichbarer Performance zu Full Fine-Tuning, dreifach reduzierter

GPU-Memory-Bedarf, und 30-prozentige Reduktion der Trainingszeit [25].

Die modulare Architektur bietet signifikante Deployment-Vorteile: Ein einzelnes Basismodell kann mit verschiedenen aufgabenspezifischen Adaptern kombiniert werden. Ein kritischer Vorteil liegt in der Möglichkeit des Weight Merging nach dem Training: Die trainierten Matrizen werden zu $W_{final} = W_0 + BA$ zusammengeführt, wodurch während der Inferenz kein zusätzlicher Overhead entsteht [25].

Trotz dieser substanziellen Vorteile weist LoRA mehrere bedeutende Limitierungen auf. Die Konvergenzproblematik stellt eine zentrale Herausforderung dar: LoRA konvergiert signifikant langsamer als Full Fine-Tuning und benötigt empirisch fünf- bis sechsmal mehr Iterationen [26]. Die Rank-Sensitivität erfordert sorgfältige Hyperparameter-Auswahl, wobei suboptimale Wahl kostspieliges Retraining erfordert. Der Performance-Gap zu Full Fine-Tuning verschärft sich bei komplexen Datensätzen mit diversen Sub-Domänen.

Eine wesentliche Erweiterung stellt Quantized Low-Rank Adaptation dar, das LoRA-Adapter mit 4-bit-Quantisierung der Basisgewichte kombiniert [27]. Diese optimierte Implementation ermöglicht eine weitere Speicherreduktion von circa 70 Prozent ohne Qualitätsverlust und macht 7-Milliarden-Parameter-Modelle auf GPUs mit weniger als 24 GB VRAM trainierbar. Die detaillierte Diskussion der Quantisierungstechniken erfolgt in Abschnitt 2.3.2.

Die Konfiguration von LoRA erfordert die Festlegung mehrerer Hyperparameter. Der Rang bestimmt die Dimensionalität der Low-Rank-Matrizen, wobei typische Werte 4, 8, 16, 32 oder 64 betragen. Der Alpha-Parameter fungiert als Skalierungsfaktor für die LoRA-Updates. Die Target Modules definieren, welche Transformer-Layer angepasst werden, typischerweise Query-, Key-, Value- und Output-Projektionen sowie bei der Mistral-Architektur zusätzlich Gate-, Up- und Down-Projektionen [25].

Adapter-Methoden stellen eines der frühesten Parameter-Efficient Fine-Tuning-Frameworks dar und folgen dem Paradigm des Additive Fine-Tuning. Der Kernansatz besteht darin, kleine aufgabenspezifische Module mit Feedforward-Layern in das gefrorene Basismodell einzufügen [22]. Adapter erfordern lediglich 3,6 Prozent zusätzliche Parameter pro Task, verursachen jedoch strukturbedingte Inferenz-Latenz durch die zusätzlichen Layer.

Prefix Tuning und Prompt Tuning repräsentieren attention- bzw. input-basierte PEFT-Ansätze. Diese Methoden optimieren kontinuierliche, aufgabenspezifische Vektoren, die den Input-Embeddings vorangestellt werden, und erreichen extreme Parameter-Effizienz [28]. Die zentrale Limitation liegt im Inferenz-Overhead durch zusätzliche Kontext-Tokens und hoher Sensitivität gegenüber der Initialisierung.

LoRA erzielt on-par oder bessere Performance als Full Fine-Tuning und übertrifft Prompt Tuning in Gesamtperformance, Memory-Effizienz und Flexibilität [24].

Adapter zeigen konsistent starke Performance auf Benchmarks, leiden jedoch unter Inferenz-Latenz. Prefix Tuning demonstriert Superiorität in multilingualen Adaptations-Tasks, während Prompt Tuning Schwächen bei kleinen Modellen zeigt.

Die Wahl von LoRA für diese Arbeit begründet sich durch sechs Faktoren: erstens kein Inferenz-Overhead durch Weight Merging, kritisch für Produktionsumgebung. Zweitens optimale Balance zwischen Parameter-Effizienz und Performance. Drittens ermöglicht Modularität mehrere Task-Adapter. Viertens Hardware-Feasibility durch Kombination mit QLoRA für Consumer-GPUs. Fünftens bewährte Performance in ähnlichen Domänen-Adaptionen. Sechstens umfangreicher Community-Support mit etablierten Best Practices.

Auch PEFT-Methoden wie LoRA erfordern bei 7-Milliarden-Parameter-Modellen erhebliche Ressourcen. Unsloth adressiert diese Herausforderungen durch drei technische Optimierungskategorien [29]. Erstens Speichereffizienz mit 50 bis 80 Prozent Reduktion des VRAM-Bedarfs: 4-bit Quantization der Base-Model-Weights erzielt circa 70 Prozent Memory-Reduktion. Die optimierte QLoRA-Implementation kombiniert LoRA-Adapter mit 4-bit Quantization. Dies ermöglicht 7-Milliarden-Parameter-Modelle auf Consumer-GPUs ohne Qualitätsverlust.

Zweitens Training-Beschleunigung durch Flash Attention 2 und Custom Kernel Implementations in OpenAI Triton [30]. Drittens numerische Stabilität durch Gradient Clipping, Layer Normalization Calibration und Multi-Precision Support.

Die Demokratisierung des Zugangs manifestiert sich in Hardware-Accessibility auf Single-GPU statt Multi-GPU, Consumer-Grade Hardware und Free Cloud Platforms. Praktische Implikationen für diese Arbeit umfassen: Iterationsgeschwindigkeit, Hardware-Feasibility und Stabilität.

2.2.2 Prompt Engineering

Fine-Tuning realisiert permanente Modellanpassung durch Parameteränderung, während Prompt Engineering Verhaltenssteuerung ohne Gewichtsänderungen ermöglicht. Der Synergieeffekt manifestiert sich in der Kombination: Fine-Tuning etabliert domänenspezifische Fähigkeiten, Prompts steuern aufgabenspezifische Kontrolle [31].

Zero-Shot Prompting beschränkt sich auf reine Instruktion ohne Demonstrationen. Base Models wie LeoLM-7B zeigen jedoch schwache Zero-Shot-Fähigkeiten. Few-Shot Learning integriert Aufgabenbeschreibung und demonstrative Beispiele im Prompt [2]. Der Mechanismus ist In-Context Learning: Das Modell erkennt Muster aus Input-Output-Paaren ohne Gewichtsanpassung. Empirische Evidenz dokumentiert dramatische Leistungssteigerung gegenüber Zero-Shot.

Die Beispielauswahl folgt spezifischen Prinzipien: Diversität gewährleistet Abde-

ckung verschiedener Störungstypen. Qualität dominiert Quantität, drei perfekte Beispiele übertreffen zehn durchschnittliche. Die Beispielreihenfolge berücksichtigt Recency Bias: Letzte Beispiele haben den stärksten Einfluss auf das Modellverhalten [32].

Instruction Prompting definiert sich als explizite Aufgabenbeschreibung vor den Beispielen. Effektive Instructions umfassen Rollenspezifikation, präzise Aufgabendefinition, Constraints bezüglich unerwünschten Verhaltens, Stilrichtlinien und Ausgabeformat [10].

Template-Strukturen umfassen vier Komponenten: System Prompt, Few-Shot Examples, User Prompt und optional Knowledge Context. Die Differenzierung zwischen Base Models und Instruction-Tuned Models ist essenziell: Base Models wie LeoLM-7B zeigen schwache Instruktionsbefolgung ohne Fine-Tuning, tendieren zur Textkontinuation statt Aufgabenlösung und benötigen Few-Shot Examples essentiell [33].

Der Synergieeffekt mit Fine-Tuning manifestiert sich: Fine-Tuning lernt domänenspezifisches Vokabular und Muster, Prompts steuern aufgabenspezifische Nuancen. Die Kombination ist robuster als jede Methode einzeln und reduziert Overfitting-Risiko durch flexible Steuerung.

2.2.3 Datensätze für Fine-Tuning

Qualität und Zusammenstellung von Trainingsdaten konstituieren kritische Erfolgsfaktoren beim Fine-Tuning. Besondere Herausforderungen manifestieren sich bei kleinen domänenspezifischen Datensätzen, wie sie in dieser Arbeit vorliegen [34]. Transfer Learning reduziert den Datenbedarf substanziell, ausreichende Qualität bleibt jedoch essentiell für erfolgreiche Adaption.

Supervised Fine-Tuning erfordert strukturierte Input-Output-Paare. Format-Anforderungen variieren zwischen JSON, JSONL und modellspezifischen Formaten. Konsistenz in Struktur und Formatierung ist obligatorisch. Prompt-Template-basiertes Format-Design folgt dem Prinzip, dass Training-Daten das Inference-Format widerspiegeln sollten [35].

Data Augmentation zielt auf Vergrößerung des Datensatzes ohne zusätzliche manuelle Annotation. Die Ziele umfassen Verbesserung der Generalisierung, Reduktion von Overfitting und Erhöhung der Robustheit [36]. Drei methodische Kategorien existieren.

Rule-Based Augmentation umfasst Synonym Replacement, Random Insertion, Random Swap und Random Deletion. Easy Data Augmentation kombiniert diese Techniken und ist effektiv bei kleinen Datensätzen [36]. Die empirische Evaluation zeigt, dass bereits einfache regelbasierte Augmentierungstechniken die Performance deutlich steigern können.

Model-Based Augmentation nutzt Back-Translation, Paraphrasing mit vortrainierten Modellen und template-basierte Generierung. Diese Methoden bieten höhere Qualität bei erhöhtem Rechenaufwand [37].

Synthetic Data Generation erzeugt komplett neue Beispiele durch LLMs. Constraint-basierte Generation mit validierten Entitäten ermöglicht Kontrolle über Diversität und Abdeckung [38]. Das Risiko liegt in Halluzinationen und unrealistischen Beispielen. Best Practice empfiehlt Kombination aus realen und synthetischen Daten.

Empirische Erkenntnisse zeigen: EDA ermöglicht mit 50 Prozent der Daten gleiche Accuracy wie 100 Prozent ohne Augmentierung [36]. Der Effekt verstärkt sich bei kleineren Datensätzen. Qualität dominiert Quantität bei synthetischen Daten.

Unbalancierte Datensätze verursachen systematische Probleme: Modelle tendieren zu häufigen Klassen und zeigen schlechtere Performance auf seltenen aber wichtigen Fällen. Strategien für Balance umfassen Oversampling unterrepräsentierter Kategorien, SMOTE-ähnliche Ansätze und class-weighted Loss Functions [39]. Empfohlen ist ein maximales Verhältnis von 1:3 zwischen seltenster und häufigster Kategorie.

Als klein gelten Datensätze mit typischerweise weniger als 1000 Beispielen für Fine-Tuning. Data Efficiency Techniques umfassen aggressive Augmentierung unter Wahrung der Qualität, parameter-effiziente Methoden wie LoRA zur Reduktion des Overfitting-Risikos, niedrigere Learning Rates und Early Stopping basierend auf Validation Loss [40].

Mixed Task Training kombiniert domänenspezifische und allgemeine Daten. Das Verhältnis liegt typisch bei 85 bis 90 Prozent spezifisch, 10 bis 15 Prozent allgemein. Dies verhindert Catastrophic Forgetting und erhält Generalisierungsfähigkeit [41].

Standard-Aufteilung: 70 bis 80 Prozent Training, 10 bis 15 Prozent Validation, 10 bis 15 Prozent Test. Bei kleinen Datensätzen: 80-10-10 oder Cross-Validation. Stratified Split bei kategorischen Daten erhält Balance. Vermeidung von Data Leakage zwischen Splits ist kritisch.

Qualitätssicherung umfasst automatisierte Validierung, manuelle Stichprobenprüfung und iterative Verbesserung basierend auf Modellfehlern [42].

2.2.4 Herausforderungen beim Fine-Tuning

Fine-Tuning vortrainierter Modelle manifestiert spezifische Herausforderungen im Spannungsfeld zwischen Spezialisierung auf neue Aufgaben und Erhalt allgemeiner Fähigkeiten. Besonders kritisch sind diese bei kleinen domänenspezifischen Datensätzen [43].

Overfitting manifestiert sich, wenn das Modell Trainingsdaten auswendig lernt statt Muster zu generalisieren. Charakteristische Symptome sind hohe Training Accuracy bei niedriger Validation bzw. Test Accuracy. Besonders ausgeprägt ist Overfitting bei kleinen Datensätzen. LoRA reduziert das Overfitting-Risiko gegenüber Full Fine-Tuning, eliminiert es jedoch nicht [25].

Die Ursachen umfassen: Modellkomplexität übersteigt Informationsgehalt der Trainingsdaten, zu viele Trainings-Epochen, unbalancierte Datensätze und zu hohe Learning Rate. Bei LoRA-Fine-Tuning spezifisch: Höherer Rank erhöht trainierbare Parameter und damit Overfitting-Risiko.

Vermeidungsstrategien gliedern sich in fünf Kategorien: Regularisierung nutzt L2-Regularisierung zur Bestrafung großer Gewichtswerte und Dropout [44]. Early Stopping monitort kontinuierlich Validation Loss während Training und stoppt Training wenn Validation Loss stagniert oder ansteigt [45]. Datenaugmentierung vergrößert den effektiven Datensatz künstlich und reduziert Overfitting durch erhöhte Variabilität. Cross-Validation nutzt K-Fold Cross-Validation bei sehr kleinen Datensätzen für robustere Schätzung der Generalisierungsfähigkeit. PEFT-spezifische Strategien: Niedrigerer LoRA-Rank reduziert trainierbare Parameter, selektive Target Modules passen nur kritische Layer an.

Catastrophic Forgetting beschreibt das Phänomen, dass Modelle während Fine-Tuning Fähigkeiten aus der Pretraining-Phase verlieren [15]. Spezialisierung auf neue Aufgaben geht zu Lasten allgemeiner Kompetenzen. Bei LoRA ist dies weniger ausgeprägt als bei Full Fine-Tuning, aber vorhanden.

Manifestationen umfassen Wissensverlust, Sprachkompetenz-Degradation und Task Interference. Gegenmaßnahmen umfassen niedrigere Learning Rates, LoRA als sanftere Alternative zu Full Fine-Tuning und Mixed Task Training [41].

LoRA zeigt inhärente Konvergenz-Limitierungen: Langsame Konvergenz erfordert fünf- bis sechsmal mehr Iterationen als Full Fine-Tuning [26]. Die Low-Rank-Constraint limitiert die Expressivität der Gewichtsanpassungen. Trade-off zwischen Rank und Konvergenzgeschwindigkeit ist omnipräsent.

Performance-Gap zu Full Fine-Tuning ist konsistent dokumentiert, besonders ausgeprägt bei komplexen Datensätzen mit diversen Sub-Domänen. Der Gap verringert sich mit größeren Basismodellen, höherem LoRA-Rank und längeren Trainings-Episoden.

2.3 Ressourceneffizienz und Modelloptimierung

Die in Abschnitt 2.2 dargestellten Fine-Tuning-Methoden ermöglichen die gezielte Anpassung von Large Language Models an spezifische Anwendungsdomänen. Während diese Verfahren die technische Grundlage schaffen, stellt sich unmittelbar die Frage nach der praktischen Realisierbarkeit: Wie können LLMs res-

sourceneffizient in produktiven Umgebungen deployed werden, insbesondere auf Consumer-Hardware mit begrenzten Ressourcen [46].

Das Spannungsfeld zwischen Modellperformance und Ressourcenverbrauch prägt die praktische Anwendung von Large Language Models in erheblichem Maße. Die Balance zwischen Qualität und Effizienz berührt dabei nicht nur technische, sondern auch wirtschaftliche und ökologische Aspekte der Modellentwicklung und -nutzung [47].

Die Ressourceneffizienz lässt sich in drei wesentliche Dimensionen untergliedern: Computational Resources umfassen die hardwareseitigen Anforderungen an Arbeitsspeicher, Rechenleistung und spezialisierte Hardware-Komponenten. Environmental Impact bezieht sich auf den Energieverbrauch und die damit verbundene CO2-Bilanz. Economic Barriers manifestieren sich in den Kosten für Training, Inferenz und Deployment von Sprachmodellen [48].

2.3.1 Problematik großer Sprachmodelle

Große Sprachmodelle sind mit fundamentalen Ressourcenproblemen konfrontiert. Die Skalierung auf Milliarden von Parametern schafft signifikante Barrieren für Deployment und praktische Nutzung [2].

Moderne große Sprachmodelle erfordern außergewöhnliche Speicherkapazitäten. GPT-3 mit 175 Milliarden Parametern benötigt über 326 Gigabyte Speicher im FP16-Format [2]. Diese Anforderung überschreitet die Kapazität selbst fortgeschrittener Grafikprozessoren bei weitem. Der zusätzliche Speicheraufwand durch Backpropagation zur Gradientenberechnung stellt insbesondere für das Training auf Endgeräten ein erhebliches Problem.

Die hohe Rechenkomplexität zeigt sich in Training und Inferenz. Bei verteiltem Training entsteht zusätzliche Komplexität durch die Vielzahl konfigurierbarer Strategien [49]. Konventionelle Optimierungsalgorithmen benötigen massive Rechenressourcen und weisen mangelnde Flexibilität bei der Parameteranpassung auf.

Die signifikanten Ressourcenanforderungen machen Cloud-Hosting häufig notwendig. Dies erfordert umfangreiche GPU-Cluster und verursacht substanzielle Kosten im vier- bis fünfstelligen Bereich. Für Forschende und kleine Organisationen entstehen dadurch erhebliche Zugangsbeschränkungen [48].

Training und Deployment großer Sprachmodelle sind aufgrund ihrer Größe und Komplexität energieintensiv. Das Training von GPT-3 verbrauchte etwa 700.000 Liter Frischwasser für die Wasserkühlung und verursachte Kosten von über 100 Millionen US-Dollar [50]. Die rasche Verbreitung großer Sprachmodelle führt zu signifikantem Energieverbrauch und CO2-Emissionen. Projektionen zeigen, dass der Energieverbrauch von Rechenzentren bis 2030 etwa 9,1 bis 11,7 Prozent des gesamten Energiebedarfs der Vereinigten Staaten ausmachen könnte [47].

In der Literatur wird zwischen Red AI und Green AI unterschieden [46]. Ein Paradigmenwechsel hin zu nachhaltiger Entwicklung erscheint erforderlich. Die Forschung zu energieeffizienten Ansätzen im maschinellen Lernen hat gezeigt, dass durch gezielte Optimierungen substantielle Reduktionen des Ressourcenbedarfs möglich sind.

Das Training großer Sprachmodelle erfordert signifikante Rechenressourcen und umfangreiche Datensätze. Dies führt zu eskalierenden Forschungs- und Entwicklungskosten [48]. Die hohen Rechenkosten führen dazu, dass fortgeschrittene künstliche Intelligenz nur für gut finanzierte Organisationen verfügbar ist.

Die finanzielle Belastung beschränkt sich nicht auf das initiale Training, sondern umfasst auch laufende Betriebskosten. In industriellen Deployments entstehen exorbitante Betriebskosten durch explosiven Overhead bei API-Aufrufen.

Cloud-Deployment wirft Datenschutzbedenken auf, verursacht Latenzprobleme und führt zu Nutzungsbeschränkungen. Edge-Deployment steht vor inhärenten Beschränkungen durch eingeschränkte Rechenleistung sowie begrenzten Speicher [51].

Für den Anwendungsfall kommunaler Organisationen ergeben sich spezifische Implikationen. Kommunale Einrichtungen verfügen über begrenzte Budgets und Ressourcen. Die Anforderungen der Datenschutz-Grundverordnung favorisieren lokales Deployment gegenüber Cloud-Lösungen. Die Beschränkung auf Consumer-Hardware schließt Enterprise-Grade-Infrastruktur aus. Diese Problemstellung erfordert Lösungsansätze, die eine effiziente Nutzung großer Sprachmodelle unter den genannten Rahmenbedingungen ermöglichen.

2.3.2 Quantisierung

Quantisierung bezeichnet die Reduktion der numerischen Präzision von Modellparametern und stellt eine zentrale Optimierungstechnik zur Effizienzsteigerung großer Sprachmodelle dar [52]. Das Grundprinzip besteht in der Umwandlung von 32-Bit-Gleitkommazahlen zu niedrigpräzisen 8-Bit- oder 4-Bit-Ganzzahldarstellungen. Diese Transformation verfolgt einen dualen Zweck: die Reduktion des Speicherbedarfs und die Beschleunigung der Berechnungen. Neuronale Netze erweisen sich als überparametrisiert und tolerieren daher einen gewissen Präzisionsverlust ohne signifikante Leistungseinbußen.

Die Quantisierung kann unterschiedlich auf Gewichte und Aktivierungen angewendet werden. Bei W8A8 werden sowohl Gewichte als auch Aktivierungen mit 8 Bit dargestellt. W4A16 nutzt 4-Bit-Gewichte bei 16-Bit-Aktivierungen, während W4A8 4-Bit-Gewichte mit 8-Bit-Aktivierungen kombiniert [53]. Die Quantisierung von Gewichten gestaltet sich einfacher als die von Aktivierungen, da letztere höhere Varianz und Ausreißer aufweisen.

Die Darstellung in 32-Bit Floating Point erfordert 4 Bytes pro Parameter,

während 8-Bit Integer lediglich 1 Byte benötigt. Dies resultiert in einer vierfachen Kompression [54]. Empirische Untersuchungen zeigen eine Reduktion der Rechenkosten um 40 Prozent und der Modellgröße um 68 Prozent. Die Leistung bleibt dabei innerhalb von 6 Prozent der FP32-Baseline.

INT8-Quantisierung nutzt ausschließlich Ganzzahlarithmetik, was zu einer Geschwindigkeitssteigerung um den Faktor 2 bis 3 führt. INT8-GEMM-Operationen erweisen sich als 1,67-mal schneller als FP32 [54]. Auf Edge-Geräten zeigt sich eine Durchsatzsteigerung um den Faktor 2,4.

INT8-Quantisierung reduziert den Energieverbrauch um 40 Prozent. Die Framerate verbessert sich um 27 Prozent, während die Inferenzzeit um 19 Prozent sinkt. Diese Eigenschaften erweisen sich als essenziell für mobile, eingebettete und Edge-Plattformen.

Theoretisch ermöglicht 4-Bit Integer-Quantisierung eine 16-fache Reduktion des Speicherbedarfs, praktisch werden jedoch Faktoren zwischen 4 und 8 erreicht [55]. Moderne Methoden realisieren eine Speicherreduktion um den Faktor 5,6. Dies ermöglicht die Ausführung von Modellen mit 7 Milliarden Parametern auf Consumer-Laptops.

INT4-Quantisierung führt zu einer Beschleunigung um den Faktor 1,45 gegenüber INT8. Mixed-Precision-Ansätze mit INT4 und INT8 reduzieren die Latenz um 23 Prozent gegenüber reiner INT8-Quantisierung.

Post-Training-INT4-Quantisierung steht jedoch vor Schwierigkeiten, die ursprüngliche Modellqualität zu erreichen [55]. Das Risiko für Genauigkeitseinbußen ist bei INT4 höher als bei INT8. GPTQ erfordert eine ausgefeilte Kalibrierung zur Erhaltung der Genauigkeit.

Die Überparametrisierung vieler neuronaler Netze ermöglicht einen minimalen Genauigkeitsverlust bei Quantisierung. INT8-Quantisierung verursacht einen Genauigkeitsverlust von unter 1 Prozent bei vierfacher Kompression [52]. INT4-Quantisierung stellt höhere Anforderungen an die Genauigkeitserhaltung.

Die Quantisierung von Aktivierungen erweist sich als anfälliger für Qualitätsverluste als die Quantisierung von Gewichten. Ausschließliche Gewichtsquantisierung erhält die Präzision der Aktivierungen und gewährleistet numerische Stabilität.

Post-Training-Quantisierung zeichnet sich durch schnelle Ausführung und Ressourceneffizienz aus, weist jedoch höhere Genauigkeitseinbußen auf. Quantization-Aware Training erhält die Genauigkeit besser, verursacht jedoch höheren Aufwand [54]. QLoRA kombiniert ein 4-Bit-Basismodell mit höherpräzisen LoRA-Adaptern [27]. Dieser hybride Ansatz verbindet Speichereffizienz während der Feinabstimmung mit Qualitätserhaltung.

INT8-Quantisierung lässt sich unkompliziert deployen und verfügt über gute Framework-Unterstützung. INT4-Quantisierung steht vor begrenzter Software-Infrastruktur. Die Leistungsgewinne hängen von Hardware-Faktoren wie Speicher-

bandbreite, Cache und Befehlssatzunterstützung ab [52].

Für den Anwendungsfall dieser Arbeit ergibt sich folgende Bewertung: INT8-Quantisierung bietet einen optimalen Kompromiss mit vierfacher Speicherreduktion, zwei- bis dreifacher Geschwindigkeitssteigerung und Genauigkeitsverlusten unter 1 Prozent. INT4-Quantisierung ermöglicht zusätzliche Einsparungen um den Faktor 5,6, birgt jedoch höhere Genauigkeitsrisiken. QLoRA kombiniert ein 4-Bit-Basismodell mit höherpräzisen Adaptern und balanciert Effizienz mit Qualität.

2.3.3 Weitere Optimierungsansätze

Neben Fine-Tuning und Quantisierung existieren weitere Ansätze zur Optimierung von Large Language Models für spezifische Anwendungsdomänen.

Das Konzept der Retrieval-Augmented Generation stellt einen hybriden Ansatz dar, der die generativen Fähigkeiten von Large Language Models mit externen Wissensdatenbanken kombiniert [56]. Die grundlegende Idee besteht darin, relevante Informationen dynamisch aus einer strukturierten Wissensbasis abzurufen und diese dem Sprachmodell als Kontext zur Verfügung zu stellen.

Der technische Ablauf unterteilt sich in zwei Phasen. In der Retrieval-Phase wird eine semantische Suche in einer Vektorendatenbank durchgeführt. In der Generation-Phase werden die abgerufenen Informationen zusammen mit der ursprünglichen Anfrage an das Large Language Model übergeben.

Die Vorteile manifestieren sich in effizienter Skalierung mit der Größe der Wissensbasis und kontinuierlicher Aktualisierbarkeit ohne erneutes Training [56]. Die Implementierung bringt jedoch erhebliche Herausforderungen mit sich. Die technische Komplexität ist deutlich höher als bei einfacheren Ansätzen, da neben dem Sprachmodell zusätzliche Komponenten wie Embedding-Modelle und Vektorendatenbanken betrieben werden müssen.

Als praktikable Alternative hat sich das Knowledge-Enhanced Prompting etabliert. Dieser Ansatz verzichtet auf dynamische Retrieval-Mechanismen und integriert stattdessen relevantes Domänenwissen statisch in die System-Prompts [57]. Die Implementierung gestaltet sich deutlich unkomplizierter als bei RAG-Systemen. Domänenspezifisches Wissen wird direkt in den Prompt eingebettet, ohne dass zusätzliche Infrastruktur erforderlich ist.

Die Menge an Wissen ist jedoch durch das Context-Window des Sprachmodells begrenzt. Zudem fehlt die Dynamik eines RAG-Systems: Aktualisierungen erfordern manuelle Anpassungen der Prompts. Für Anwendungsfälle im Verkehrssektor eignet sich Knowledge-Enhanced Prompting besonders dann, wenn die relevante Wissensbasis überschaubar und relativ stabil ist.

2.4 Qualitätssicherung bei KI-generierten Texten

Die in den Abschnitten 2.2 und 2.3 diskutierten Optimierungsverfahren schaffen die technische Grundlage für die Anpassung von Large Language Models an spezifische Anwendungsdomänen. Damit stellt sich jedoch eine entscheidende Frage: Wie kann sichergestellt werden, dass die generierten Texte qualitativ hochwertig sind und den Anforderungen einer produktiven Anwendung genügen [58].

Die Qualitätssicherung bei LLM-generierten Texten stellt besondere Herausforderungen dar. Während deterministische Systeme bei identischen Eingaben reproduzierbare Ausgaben liefern, ist das Verhalten generativer Modelle durch ihre stochastische Natur geprägt. Dieser Non-Determinismus birgt das Risiko inkonsistenter Outputs und erfordert spezifische Evaluationsstrategien [19].

Im Kontext von Verkehrsinformationen ist die Faktentreue nicht optional, sondern essentiell. Falsche oder inkonsistente Angaben zu Liniennummern, Haltestellen oder Betriebszeiten können direkte Auswirkungen auf die Reiseplanung der Fahrgäste haben. Die Balance zwischen automatisierter Evaluation und manueller Qualitätskontrolle stellt dabei eine zentrale methodische Herausforderung dar.

Halluzinationen bezeichnen das Phänomen, dass Sprachmodelle plausibel klingende, aber faktisch falsche Informationen generieren [59]. Besonders problematisch ist dabei, dass halluzinierte Informationen oft syntaktisch und stilistisch korrekt formuliert sind und sich nur durch Abgleich mit den strukturierten Eingabedaten als fehlerhaft identifizieren lassen.

Inkonsistenz resultiert aus der stochastischen Natur von LLMs und manifestiert sich in variierenden Outputs bei identischen Inputs. Der Trade-off zwischen Kreativität und Determinismus muss durch geeignete Hyperparameter wie die Temperature gesteuert werden [60].

Stilabweichungen treten auf, wenn generierte Texte von etablierten Formulierungsrichtlinien abweichen. Unvollständigkeit bezeichnet das selektive Weglassen wichtiger Informationen aus der Eingabe. Insbesondere bei sicherheitsrelevanten Angaben ist ein Informationsverlust durch Zusammenfassung kritisch.

Automatische Metriken ermöglichen eine skalierbare Evaluation großer Textmengen, sind jedoch in ihrer Aussagekraft limitiert [61]. Oberflächliche Metriken wie BLEU und ROUGE messen lediglich N-gram Overlaps. Semantische Metriken wie BERTScore beruhen auf Embedding-basierten Vergleichen.

Für domänenspezifische Anforderungen sind spezialisierte Validierungsverfahren erforderlich. Self-Consistency-Ansätze generieren mehrere Outputs für denselben Input und bilden einen Konsens [62]. Fact-Checking-Verfahren validieren die generierten Texte gegen strukturierte Eingabedaten. Prompt-basierte Präventionsstrategien setzen bereits bei der Generierung an, indem sie Cons-

traints im System Prompt definieren.

2.4.1 Evaluationsmetriken für NLP

Die Evaluation von Textgenerierungsmodellen erfordert geeignete Metriken zur Quantifizierung der Ausgabequalität. Im Folgenden werden die drei zentralen Metriken BLEU, ROUGE und BERTScore erläutert.

BLEU wurde für die Evaluation maschineller Übersetzungssysteme entwickelt und basiert auf dem Vergleich von N-Gramm-Übereinstimmungen zwischen generiertem Text und Referenztext [63]. Die Metrik berechnet die Präzision übereinstimmender N-Gramme verschiedener Längen und kombiniert diese mittels geometrischem Mittel. Die Stärke liegt in der schnellen Berechenbarkeit und der Unabhängigkeit von menschlichen Annotatoren. Allerdings erfasst BLEU lediglich oberflächliche Textübereinstimmungen und kann semantische Äquivalenz nicht adäquat bewerten.

ROUGE wurde für die automatische Evaluation von Textzusammenfassungen entwickelt [64]. ROUGE-N misst die Übereinstimmung von N-Grammen, wobei im Gegensatz zu BLEU der Recall betont wird. ROUGE-L basiert auf der längsten gemeinsamen Teilsequenz zwischen generiertem und Referenztext. Die Fokussierung auf Recall macht ROUGE besonders geeignet für die Evaluation von Zusammenfassungen, bei denen die Vollständigkeit der Informationsabdeckung zentral ist.

BERTScore adressiert die Limitationen oberflächlicher N-Gramm-Metriken durch die Nutzung kontextualisierter Worteinbettungen [65]. Die Metrik berechnet die Kosinus-Ähnlichkeit zwischen den BERT-Embeddings der Token im generierten und im Referenztext. Die Verwendung kontextualisierter Embeddings ermöglicht die Erfassung semantischer Äquivalenz jenseits exakter lexikalischer Übereinstimmungen. BERTScore korreliert stärker mit menschlichen Qualitätsurteilen.

2.4.2 Halluzination Detection und Validierung

Halluzinationen bezeichnen die Generierung faktisch inkorrekter oder erfundener Informationen durch Sprachmodelle. Diese Problematik erweist sich als besonders kritisch bei sicherheitsrelevanten Verkehrsinformationen [19].

Self-Consistency beschreibt einen Ansatz zur Qualitätssicherung durch mehrfache Generierung und Konsensanalyse [62]. Das Grundprinzip besteht darin, für dieselbe Eingabe mehrere verschiedene Outputs zu generieren und diese auf Übereinstimmung zu prüfen.

Der Mechanismus umfasst die Generierung von N verschiedenen Outputs für identische Eingaben, typischerweise N zwischen 3 und 10. Das Sampling er-

folgt mit einer Temperatur größer null, um Variabilität in den Ausgaben zu ermöglichen. Die zugrunde liegende Annahme besagt, dass korrekte Antworten konvergieren, während Halluzinationen divergieren.

Die Limitation besteht im N-fachen Rechenaufwand durch mehrfache Inferenz. Für die vorliegende Arbeit wird ein modifizierter Self-Consistency-Ansatz implementiert: Eine erste Instanz generiert den initialen Output mit niedriger Temperatur. Eine zweite Instanz desselben Modells prüft diesen Output anhand expliziter Validierungskriterien und korrigiert identifizierte Fehler.

Die strukturierte Validierung generierter Texte erfolgt durch Extraktion und Vergleich von Entitäten [58]. Aus Input und Output werden relevante Entitäten extrahiert und gegenübergestellt. Eine fundamentale Regel besagt, dass der Output keine Entitäten enthalten darf, die nicht im Input vorhanden sind.

Named Entity Recognition ermöglicht die Extraktion spezifischer Informationen wie Liniennummern, Haltestellen, Zeitangaben und Fahrzeugtypen. Dies kann mittels Pattern-Matching oder dedizierter NER-Modelle erfolgen. Constraint-basierte Validierung nutzt einen Whitelist-Ansatz, bei dem ausschließlich bekannte Liniennummern zugelassen werden.

Prompt-Engineering bietet Mechanismen zur präventiven Reduktion von Halluzinationen [66]. Negative Constraints im System-Prompt explizieren unerwünschte Verhaltensweisen. Typische Formulierungen umfassen die strikte Anweisung, keine Details zu erfinden, die nicht in der Eingabe enthalten sind, und ausschließlich Informationen aus dem bereitgestellten Kontext zu nutzen.

Knowledge Grounding beschreibt die explizite Einbettung von Fakten im Prompt. Diese Technik reduziert Halluzinationen durch direkte Verfügbarkeit relevanten Wissens im Kontext.

Die Temperatur-Parameter der Generierung beeinflusst die Zufälligkeit der Token-Selektion. Eine Temperatur nahe null induziert deterministische Generierung, bei der stets die wahrscheinlichsten Token gewählt werden [60]. Dies reduziert Halluzinationen, da unwahrscheinliche und potenziell inkorrekte Token-Sequenzen vermieden werden.

Der Trade-off besteht in reduzierter Kreativität bei erhöhter Konsistenz. Für faktische Transformationen, bei denen Korrektheit gegenüber Variabilität priorisiert wird, erweist sich dieser Ansatz als geeignet. In der vorliegenden Arbeit wird die Temperatur auf null gesetzt, um maximale Determinismus und Reproduzierbarkeit zu gewährleisten.

Automatisierte Evaluationsverfahren bieten Skalierbarkeit, weisen jedoch Limitationen auf. Entitätsvergleich erfolgt schnell und deterministisch. Pattern-Matching basiert auf Regeln, erfordert jedoch kontinuierliche Wartung.

Manuelle Evaluation durch menschliche Annotatoren stellt den Gold Standard dar, skaliert jedoch nicht für große Datenmengen [61]. Ein hybrider Ansatz

kombiniert die Vorteile beider Methoden. Automatisierte Pre-Filter fangen offensichtliche Fehler ab. Manuelle Review fokussiert auf Grenzfälle und Stichproben. Dieser Ansatz erweist sich als optimal für praktische Anwendungen unter Ressourcenbeschränkungen.

Die theoretischen Konzepte dieses Abschnitts bilden die methodische Grundlage für die praktische Evaluation. Die konkrete Anwendung der beschriebenen Techniken sowie die Messung ihrer Effektivität anhand spezifischer Metriken erfolgt in Kapitel 7.

3 Anwendungskontext und Anforderungen

3.1 Analyse der Verkehrsanweisungen

Die Leipziger Verkehrsbetriebe erstellen technische Verkehrsanweisungen, um interne Abläufe bei Baumaßnahmen, Umleitungen oder Betriebsänderungen zu koordinieren. Diese Dokumente dienen primär der internen Kommunikation zwischen verschiedenen Abteilungen und enthalten sowohl fahrgastrelevante als auch rein betriebliche Informationen. Die folgenden Abschnitte analysieren die Struktur dieser Anweisungen sowie die daraus abgeleiteten Anforderungen an die Transformation in nutzergerechte Fahrgastinformationen.

3.1.1 Kategorisierung von Verkehrsmeldungen nach Komplexität

Die Komplexität von Verkehrsanweisungen wird durch verschiedene strukturelle Dimensionen bestimmt, die sich auf die Anforderungen an die automatisierte Transformation auswirken [67]. Diese Dimensionen treten in der Praxis häufig in Kombination auf, wobei umfangreichere Verkehrsanweisungen typischerweise mehrere dieser Merkmale gleichzeitig aufweisen, während einfachere Anweisungen sich auf einzelne Aspekte beschränken können. Die Systematisierung von Komplexitätsdimensionen in technischen Dokumenten folgt etablierten Ansätzen der Informationsarchitektur und Dokumentenanalyse [68], [69].

Eine grundlegende Dimension der Komplexität stellt die Anzahl der betroffenen Linien dar. Die einfachste Konstellation betrifft eine einzelne Linie, beispielsweise wenn eine Buslinie aufgrund einer kurzzeitigen Straßensperrung umgeleitet wird. Komplexere Szenarien umfassen mehrere Linien, die entweder identische oder unterschiedliche Maßnahmen erfahren. Bei identischen Maßnahmen können die betroffenen Linien durch Aufzählung zusammengefasst werden, etwa wenn mehrere Buslinien dieselbe Umleitungsstrecke nutzen. Deutlich anspruchsvoller gestaltet sich die Transformation, wenn jede betroffene Linie individuelle Maßnahmen erfährt.

Eine weitere Komplexitätsdimension ergibt sich aus der Richtungsspezifität der Maßnahmen. Während bei einfachen Anweisungen eine Linie lediglich in eine Fahrtrichtung betroffen ist, können komplexere Fälle unterschiedliche Maßnahmen für verschiedene Richtungen derselben Linie vorsehen. In solchen Fällen müssen die richtungsspezifischen Maßnahmen in der Fahrgastinformation klar voneinander abgegrenzt werden, wobei die Konvention besteht, die Maßnahmen sequenziell zu formulieren [70].

Die zeitliche Dimension fügt eine zusätzliche Komplexitätsebene hinzu. Einfache Verkehrsanweisungen gelten für einen einheitlichen Zeitraum. Komplexere Anweisungen können jedoch mehrere Zeiträume oder Bauphasen umfassen, in denen unterschiedliche Maßnahmen gelten. Diese zeitliche Staffelung erfordert ei-

ne präzise Zuordnung der Maßnahmen zu den jeweiligen Gültigkeitszeiträumen, wobei Fahrgäste ausschließlich die für sie relevanten Informationen erhalten sollen [71].

Die Art der verkehrlichen Maßnahmen bildet eine weitere Differenzierungsdimension. Diese reicht von einfachen Umleitungen über Haltestellenänderungen bis hin zu Ersatzverkehren, also temporären Busverbindungen die ausgefallene Bahn- oder Straßenbahnstrecken ersetzen. Besondere Herausforderungen entstehen, wenn verschiedene Maßnahmentypen kombiniert werden, beispielsweise wenn eine Linie verkürzt wird und gleichzeitig ein Schienenersatzverkehr, also ein Ersatzbus, die Reststrecke übernimmt. In solchen Fällen müssen die Zusammenhänge zwischen den Maßnahmen und die Umsteigeverbindungen explizit kommuniziert werden.

Die räumliche Ausdehnung der Maßnahmen stellt ebenfalls einen Komplexitätsfaktor dar. Während lokale Änderungen wie die Verlegung einer einzelnen Haltestelle relativ übersichtlich kommuniziert werden können, erfordern großräumige Umleitungen oder stadtweite Betriebsänderungen eine differenziertere Darstellung der betroffenen Streckenabschnitte und Haltestellenbereiche [72].

Eine besondere Komplexitätsdimension ergibt sich aus den Interdependenzen zwischen verschiedenen Maßnahmen. Hierzu zählen Substitutionen, bei denen eine Linie durch eine andere ersetzt wird, Modifikationen, bei denen eine Linie mit Änderungen weiterfährt, sowie Eliminierungen, bei denen eine Linie vollständig entfällt. Das Sprachmodell muss lernen, diese Beziehungen zu erkennen und in der Fahrgastinformation verständlich darzustellen.

Schließlich können Verkehrsanweisungen faktisch mehrere separate Baumaßnahmen bündeln, die räumlich oder zeitlich zusammenhängen, aber jeweils eigenständige Transformationen erfordern. Für die automatisierte Verarbeitung bedeutet dies, dass das System erkennen muss, welche Maßnahmen logisch zusammengehören und gemeinsam kommuniziert werden sollten.

Die beschriebenen Komplexitätsdimensionen bestimmen die Diversität des Trainingsdatensatzes und beeinflussen die Modellperformance. Umfangreiche Verkehrsanweisungen weisen typischerweise mehrere dieser Dimensionen gleichzeitig auf, während einfachere Anweisungen sich auf einzelne Aspekte beschränken können.

3.1.2 Charakterisierung technischer Verkehrsanweisungen

Verkehrsanweisungen folgen bei den LVB einer standardisierten Struktur, die sich aus der betrieblichen Praxis entwickelt hat. Die Standardisierung technischer Dokumente in Organisationen folgt etablierten Prinzipien der Organizational Communication und Document Design [73], [74]. Die Analyse der Anweisung 198/25 verdeutlicht den typischen Aufbau dieser Dokumente und dient im

Folgenden als konkretes Beispiel für die Erläuterung der Strukturelemente.

Die Grundlage für die nachfolgenden Erläuterungen bildet das Dokument Sperrung der Lützner Straße, das die Struktur und Inhalte einer typischen Verkehrsanweisung exemplarisch darstellt.

Jede Verkehrsanweisung beginnt mit einer Kopfzeile, die das LVB-Branding, Ort und Datum der Veröffentlichung, Leipzig, 24.09.2025, einen eindeutigen Titel sowie die Anweisungsnummer enthält. Darauf folgt die Nennung untergeordneter Anweisungen, sofern mehrere Baumaßnahmen zeitlich oder räumlich zusammenhängen. In der Anweisung 198/25 wird beispielsweise auf Anweisung Nr. 100/24 – Sperrung der Jahnallee verwiesen, was auf eine räumliche oder betriebliche Verbindung zwischen beiden Maßnahmen hinweist. Anschließend erfolgt die Übersicht der betroffenen Linien, in diesem Fall Linie N2, operativer SEV.

Der Hauptteil der Anweisung gliedert sich in mehrere inhaltlich voneinander abgegrenzte Abschnitte. Zunächst wird der Anlass der Anweisung erläutert: Für den Neubau der landwärtigen Bushaltestelle sowie einer Querungshilfe wird vom Montag, 13.10.2025 bis Freitag, 17.10.2025 die Lützner Straße zwischen Brünner Straße und Nelkenweg für den Kraftfahrzeug-Verkehr in beiden Richtungen gesperrt. Diese Passage kombiniert Grund, Zeitraum und räumliche Eingrenzung der Maßnahme. Dabei werden auch begleitende Maßnahmen genannt, die für das Verständnis der Gesamtsituation relevant sind.

Für jede betroffene Linie werden die spezifischen Maßnahmen detailliert beschrieben. Die Anweisung 198/25 beschreibt für Linie N2: verkehrt ab der Nacht vom Mo. 13.10./ Di. 14.10. zwischen den Haltestellen Saarländer Straße und Schönauer Ring in beiden Richtungen mit Umleitung über Brünner Straße – Ratzelstraße – Schönauer Straße – Lützner Straße und bedient alle Haltestellen entlang der Umleitung. Diese Beschreibung umfasst neue Linienführungen in Form einer Umleitung mit präziser Straßennennung sowie implizite Informationen zu Halteausfällen, wobei die regulären Haltestellen im gesperrten Bereich entfallen, während Haltestellen entlang der Umleitungsroute bedient werden.

Zusätzlich enthalten die Anweisungen Ansagetexte für Fahrzeugführer, die an bestimmten Haltestellen abgespielt werden sollen, um Fahrgäste über die Änderungen zu informieren. In der Anweisung 198/25 sind zwei richtungsspezifische Ansagetexte enthalten. Der Text für die Haltestelle Saarländer Straße nach Markranstädt lautet: Wegen Sperrung der Lützner Straße verkehrt diese Linie weiter mit Umleitung über Brünner Straße – Ratzelstraße – Schönauer Straße – Lützner Straße zur Haltestelle Schönauer Ring. Die Haltestellen Grünauer Allee und Parkallee können nicht bedient werden. Der entsprechende Text für die Gegenrichtung Schönauer Ring nach Hauptbahnhof ist analog formuliert, nennt aber die Straßen in umgekehrter Reihenfolge, entsprechend der Fahrtrichtung.

Die technischen Anweisungen richten sich an interne Fachabteilungen und umfassen beispielsweise die Schließung und Neueinrichtung von Haltestellen. Der

Abschnitt BMH, Betriebsmittel und Haltestellen, in Anweisung 198/25 enthält folgende Arbeitsaufträge: Haltestellen Parkallee und Grünauer Allee in beiden Richtungen für die Linien N2 und X5 schließen, alle Haltestellen im Umleitungsweg der Linie N2 in beiden Richtungen einrichten, Fahrgastinformationen und Fahrpläne veröffentlichen. Diese Informationen sind nicht für Fahrgäste relevant und müssen bei der Transformation herausgefiltert werden [75]. Am Ende der Anweisung befinden sich Fahrtwegskizzen, die die Umleitungen und neuen Linienführungen visuell darstellen und die textlichen Beschreibungen durch kartografische Darstellungen ergänzen.

Nicht alle in Verkehrsanweisungen enthaltenen Informationen sind für Fahrgäste relevant. Eine Analyse der Informationshierarchie zeigt, dass sich die Inhalte in zwei Hauptkategorien einteilen lassen. Die Kategorisierung von Information nach Relevanz für unterschiedliche Zielgruppen folgt etablierten Prinzipien der Informationsarchitektur und Audience Analysis [76], [77]. Die erste Kategorie umfasst fahrgastrelevante Informationen, die die Grundlage jeder Fahrgastinformation bilden und in verständlicher Form kommuniziert werden müssen. Dazu zählen der Gültigkeitszeitraum der Änderungen, vom Montag, 13.10.2025 bis Freitag, 17.10.2025, die betroffenen Linien, Linie N2, und die Art der Beeinträchtigung, mit Umleitung, sowie die räumliche Eingrenzung zwischen den betroffenen Haltestellen, zwischen Saarländer Straße und Schönauer Ring. Der Grund der Maßnahme liefert den notwendigen Kontext, Neubau der landwärtigen Bushaltestelle sowie einer Querungshilfe. Darüber hinaus müssen spezifische Haltestellenänderungen kommuniziert werden, in diesem Fall die ausfallenden Haltestellen Grünauer Allee und Parkallee sowie die bediente Umleitungsroute.

Die zweite Kategorie umfasst technische Informationen, die ausschließlich der internen Koordination dienen und Fahrgäste verwirren oder mit unnötigen Details überfrachten würden. In der Anweisung 198/25 gehört dazu beispielsweise die Erwähnung der Linie X5 im BMH-Abschnitt. Diese Linie ist nicht von der Umleitung betroffen, sondern lediglich von der Haltestellenschließung im gesperrten Bereich. Für Fahrgäste der Linie N2 ist diese Information irrelevant; Fahrgäste der Linie X5 erhalten separate Informationen. Ebenso sind Details zur Einrichtung von Haltestellen im Umleitungsweg rein operativ und müssen nicht kommuniziert werden.

Die Analyse des Korpus von Verkehrsanweisungen zeigt unterschiedliche Komplexitätsgrade gemäß der in Abschnitt 3.1.1 dargelegten Dimensionen. Eine weitere Herausforderung für die automatisierte Verarbeitung stellt die Disambiguierung, also die eindeutige Zuordnung von Linieninformationen dar. Verkehrsanweisungen enthalten nicht nur die von den Maßnahmen betroffenen Linien, sondern nennen auch Linien, die lediglich zur Orientierung oder als Kontext dienen. Die Anweisung 198/25 illustriert dies beispielhaft: Während in der Kopfzeile eindeutig Linie N2 als betroffene Linie genannt wird, erscheint im BMH-Abschnitt zusätzlich Linie X5. Für das Sprachmodell ist es essenziell, diese Unterscheidung zu treffen und nur die tatsächlich von der Umleitung betroffene Linie N2 in die

Fahrgastinformation zu übernehmen.

Bei größeren Baumaßnahmen können Verkehrsanweisungen sehr schnell an Komplexität zunehmen; dies wird beispielhaft in der Anweisung 200/25 deutlich. Das zugehörige Dokument zu den Gleisbauarbeiten in der Riesaer Straße befindet sich im Anhang.

3.1.3 Analyse bestehender Fahrgastinformationen

Um Regeln und Konventionen für die automatisierte Transformation zu entwickeln, wurden bestehende Fahrgastinformationen systematisch analysiert. Die Methodik folgt etablierten Ansätzen der Korpusanalyse und vergleichenden Dokumentenanalyse im Bereich der technischen Kommunikation [78], [79]. Die Analyse umfasste 185 Beispiele von verschiedenen deutschen Verkehrsbetrieben, darunter die Schweizerische Bundesbahnen, die Berliner Verkehrsbetriebe, den Verkehrsverbund Bremen/Niedersachsen, den Verkehrsverbund Berlin-Brandenburg und den Rhein-Main-Verkehrsverbund. Zusätzlich wurden bereits veröffentlichte Fahrgastinformationen der LVB als Referenz herangezogen.

Eine ausführliche Sammlung der analysierten Beispiele sowie die daraus abgeleiteten Formulierungsregeln sind im Dokument Regeln Formulierung Fahrgastinformationen im Anhang, siehe Anhang Abschnitt A, enthalten und dienen als zentrale Referenz für die nachfolgende Systematisierung.

Die Extraktion erfolgte auf Grundlage einer bereitgestellten Datei, in der Beispieltexte bereits nach potenziellen Satzbausteinen kategorisiert waren. Diese Kategorisierung wurde manuell überprüft und die relevanten Formulierungen markiert. Anschließend wurden die Satzbausteine unter allgemeineren Maßnahmentypen zusammengefasst, um eine hierarchische Struktur für die spätere Prompt-Entwicklung zu schaffen. Die in der Datei enthaltenen Beispiele wurden mit den tatsächlichen Verkehrsanweisungen der LVB sowie den bereits verfassten Texten abgeglichen. Nur solche Formulierungen und Regeln, die für die LVB zutreffend und konsistent anwendbar waren, wurden in das finale Regelwerk aufgenommen. Dabei wurde darauf geachtet, die Kontextlänge, also die maximale Eingabegröße des Sprachmodells, optimal auszunutzen, ohne durch zu viele Beispiele die Gefahr von Fehlern oder Halluzinationen zu erhöhen.

Aus der Analyse ergaben sich mehrere Kernerkenntnisse, die die Basis für die Entwicklung der Transformationsregeln bilden. Die folgenden Abschnitte beleuchten zentrale Konventionen, die über die analysierten Verkehrsbetriebe hinweg konsistent auftreten und für die automatisierte Verarbeitung von besonderer Bedeutung sind.

Fahrzeugtyp-Spezifikation und terminologische Unterscheidungen

Während Verkehrsanweisungen häufig generische Formulierungen wie Linie N2 verwenden, erwarten Fahrgäste eine explizite Nennung des Fahrzeugtyps in der

Form BUS N2 oder TRAM 7. Diese Konvention ist über alle analysierten Verkehrsbetriebe hinweg konsistent und erhöht die Klarheit, insbesondere wenn sowohl Bus- als auch Straßenbahnlinien mit ähnlichen Nummern existieren. Die explizite Nennung von Fahrzeugtypen in Fahrgastinformationen entspricht etablierten Prinzipien der Plain Language und technischen Kommunikation, die Eindeutigkeit und Verständlichkeit priorisieren [80], [81], [82]. Bei der gleichzeitigen Nennung mehrerer Linien desselben Fahrzeugtyps wird der Typ nur einmal vorangestellt, etwa BUS 72, 73 und N7 fahren. Diese Regel verbessert die Lesbarkeit, insbesondere bei langen Linienaufzählungen.

Eine besonders wichtige terminologische Unterscheidung betrifft die Bezeichnung von Haltepunkten. Während Busse und Straßenbahnen im Betriebsalltag häufig undifferenziert behandelt werden, folgt die Fahrgastkommunikation einer strikten Konvention: Buslinien verwenden ausschließlich Haltestelle beziehungsweise Haltestellen und Ersatzhaltestelle beziehungsweise Ersatzhaltestellen, während Bahnen und Straßenbahnen die Begriffe Halt beziehungsweise Halte und Ersatzhalt beziehungsweise Ersatzhalte nutzen. Diese Differenzierung zieht sich durch das gesamte Regelwerk und betrifft alle Maßnahmentypen. So lautet beispielsweise die Formulierung für einen Halteausfall bei einem Bus Die Haltestelle Markt entfällt ersatzlos, während bei einer Straßenbahn Der Halt Markt entfällt ersatzlos formuliert wird. Diese Unterscheidung ist nicht nur eine sprachliche Konvention, sondern spiegelt die unterschiedlichen Betriebscharakteristika wider und trägt zur Professionalität der Kommunikation bei.

Strukturelle Grundregeln und Informationsanordnung

Die Analyse bestätigte eine konsistente Grundstruktur über verschiedene Verkehrsbetriebe hinweg, die sich in vier Hauptelemente gliedert: Zunächst der Zeitraum, der angibt, wann die Änderung gilt, vom Montag, 13. Oktober bis Freitag, 17. Oktober 2025. Darauf folgen die betroffenen Linien mit explizitem Fahrzeugtyp, BUS N2. Es folgen die konkreten Maßnahmen, verkehrt zwischen den Haltestellen Saarländer Straße und Schönauer Ring in beiden Richtungen mit Umleitung über Brünner Straße, Ratzelstraße, Schönauer Straße und Lützner Straße. Den Abschluss bildet die Nennung des Grundes für die Änderung, Grund dafür ist der Neubau der landwärtigen Bushaltestelle sowie einer Querungshilfe. Diese Strukturierung folgt etablierten Prinzipien der Informationsarchitektur, die die Priorisierung handlungsrelevanter Informationen vor kontextuellen Erklärungen vorsehen [67].

Die Platzierung des Grundes am Ende wurde aus zwei Überlegungen heraus als optimal identifiziert. Erstens ist für Fahrgäste zunächst die konkrete Auswirkung auf ihre Reise von primärer Relevanz; der Grund liefert Kontext, ist aber für die unmittelbare Reiseplanung weniger entscheidend. Zweitens entspricht diese Struktur der gängigen Praxis bei anderen deutschen Verkehrsbetrieben und trägt zur Konsistenz und Wiedererkennbarkeit bei. Der Grund wird dabei stets als eigenständiger Satz formuliert: Grund dafür ist Singular-Beschreibung oder Grund dafür sind Plural-Beschreibung. Diese strikte Formulierung vermeidet

Variationen und erhöht die Vorhersagbarkeit für Fahrgäste. Bei der Verkehrsanweisung 198/25 wird beispielsweise die technische Formulierung Für den Neubau der landwärtigen Bushaltestelle sowie einer Querungshilfe wird die Lützner Straße gesperrt transformiert zu Grund dafür ist der Neubau der landwärtigen Bushaltestelle sowie einer Querungshilfe.

Formatierungskonventionen und sprachliche Details

Mehrere präzise Formatierungsregeln tragen zur Einheitlichkeit der Fahrgastinformationen bei. Haltestellennamen werden ausnahmslos in Anführungszeichen gesetzt, etwa Die Haltestelle Grünauer Allee kann nicht bedient werden. Diese Regel gilt unabhängig vom Kontext und unterscheidet Haltestellennamen klar von Straßennamen, die ohne Anführungszeichen genannt werden. Die systematische Verwendung von Anführungszeichen zur Markierung von Eigennamen entspricht etablierten typografischen Konventionen in der technischen Dokumentation [77].

Bei Aufzählungen wird das letzte Element stets mit und, nicht mit Komma, angeschlossen. Eine korrekte Aufzählung lautet Brünner Straße, Ratzelstraße, Schönauer Straße und Lützner Straße, nicht Brünner Straße, Ratzelstraße, Schönauer Straße, Lützner Straße. Diese als Oxford-Comma-Verzicht bekannte Konvention entspricht der gängigen deutschen Sprachpraxis und wird konsequent in allen Aufzählungskontexten angewendet, sowohl bei Straßennamen in Umleitungen als auch bei der Nennung mehrerer ausfallender Haltestellen.

Die Analyse der LVB-Texte zeigt einen heterogenen Umgang mit Abkürzungen. Während allgemein bekannte Straßenabkürzungen wie -str. für -straße in Fahrgastinformationen akzeptabel sind, werden fachspezifische Abkürzungen inkonsistent behandelt. Für ein standardisiertes Format wurde die Entscheidung getroffen, fachspezifische Abkürzungen konsequent auszuschreiben: Hst. wird zu Haltestelle, Strbhf. zu Straßenbahnhof und örtl. zu örtlicher. Diese Regel erhöht die Verständlichkeit für alle Fahrgäste, insbesondere für weniger verkehrsaffine Personen oder ortsunkundige Reisende. Die konsequente Auflösung von Abkürzungen entspricht den Empfehlungen für barrierefreie und verständliche Kommunikation [83], [84]. In der Verkehrsanweisung 198/25 wird die Linie N2 zwar nicht verkürzt, jedoch ist in anderen Anweisungen die Formulierung verkürzt bis Paunsdorf, Strbhf. zu finden, die in der Fahrgastinformation zu verkürzt bis und ab Paunsdorf, Straßenbahnhof transformiert wird.

Maßnahmenspezifische Regelungen

Unterschiedliche Maßnahmentypen folgen jeweils eigenen Formulierungsmustern. Bei Halteausfällen wird zwischen Singular und Plural unterschieden: Die Haltestelle Grünauer Allee kann nicht bedient werden für einen einzelnen Ausfall, versus Die Haltestellen Grünauer Allee und Parkallee können nicht bedient werden für mehrere Ausfälle. Das Verb muss entsprechend konjugiert werden. In der Verkehrsanweisung 198/25 werden beide Haltestellen explizit in den Ansagetexten genannt, was die Plural-Formulierung erfordert.

Bei Umleitungen zeigt sich ein fahrzeugtypspezifischer Unterschied. Während bei Bussen die Formulierung Die Busse fahren eine Umleitung über verwendet wird, lautet die entsprechende Formulierung bei Straßenbahnen Die Züge werden über umgeleitet. Diese Unterscheidung trägt den unterschiedlichen Betriebscharakteristika Rechnung und findet sich in den Fahrgastinformationen verschiedener deutscher Verkehrsbetriebe. In kürzeren Texten kann die vereinfachte Form mit Umleitung über verwendet werden, wie in der Verkehrsanweisung 198/25: verkehrt mit Umleitung über Brünner Straße, Ratzelstraße, Schönauer Straße und Lützner Straße.

Sonderfälle und Abweichungen

Bestimmte Situationen erfordern Abweichungen von der Grundstruktur. Bei Endmeldungen, die die Aufhebung von Einschränkungen kommunizieren, entfällt die Grund-Angabe vollständig. Die Formulierung beschränkt sich auf Zeitangabe und Meldung: ab 20.05.23: Die reguläre Haltestelle Markt ist wieder in Betrieb. Ein Grund wäre hier redundant, da die vorangegangene Einschränkung den Fahrgästen bereits bekannt ist und die Endmeldung lediglich den Normalzustand wiederherstellt.

Bei Events, Sportveranstaltungen, Messen, Lichtfest, ändert sich die Struktur dahingehend, dass der Event-Anlass nach der Zeitangabe, aber vor der Liniennennung steht: am Samstag, 18. Oktober 2025: Anlässlich des Fußballspiels RB Leipzig gegen Hamburger SV fahren die Linien. In diesem Fall kann der Grund am Ende entfallen, da der Event-Anlass bereits am Anfang genannt wurde und eine Wiederholung redundant wäre.

Ein weiterer Sonderfall sind Ansagetexte, die in Fahrzeugen abgespielt werden. Diese folgen einer eigenen Struktur: Sie beginnen stets mit Sehr geehrte Fahrgäste, und integrieren den Grund direkt mit wegen Grund nach der Begrüßung. Zeitangaben und Liniennummern entfallen vollständig, stattdessen wird der Fahrzeugbezug verwendet, dieser Bus beziehungsweise diese Straßenbahn. Die beiden in der Verkehrsanweisung 198/25 vorgegebenen Ansagetexte illustrieren diese Struktur: Wegen Sperrung der Lützner Straße verkehrt diese Linie weiter mit Umleitung über.

Systematisierung und Übertragbarkeit

Die Entwicklung eines automatisierten Transformationssystems bietet die Chance, über die reine Zeitersparnis hinaus einen systematischen Ansatz für die Fahrgastinformation zu etablieren. Während Mitarbeitende bei der manuellen Formulierung unter Zeitdruck arbeiten und individuelle Präferenzen in die Texterstellung einfließen lassen, ermöglicht die Automatisierung eine durchgängig konsistente Anwendung der definierten Regeln. Die systematische Standardisierung von Kommunikationsprozessen in Organisationen trägt nachweislich zur Qualitätssicherung und Effizienzsteigerung bei [73], [74]. Durch die systematische Anwendung werden Variationen in Formulierungen vermieden, die bei manueller Texterstellung naturgemäß auftreten. Dies erhöht die Wiedererkennbarkeit

und reduziert kognitive Belastung für Fahrgäste, die regelmäßig auf Verkehrsinformationen angewiesen sind [85], [86].

Es ist dabei hervorzuheben, dass die ausgearbeiteten Regeln nicht lediglich eine Sammlung von Beispielen anderer Verkehrsbetriebe darstellen. Vielmehr bilden sie die Grundlage für einen übergreifend einheitlichen Stil zur Formulierung von Fahrgastinformationen, auf den sich die beteiligten Betriebe verständigen. Die Regeln dienen somit als verbindlicher Rahmen, der eine konsistente und verständliche Kommunikation sicherstellt und die Grundlage für eine überregionale Standardisierung schafft.

Ein weiterer Vorteil liegt in der Möglichkeit, das entwickelte Regelwerk auch überregional nutzbar zu machen. Die identifizierten Konventionen basieren nicht nur auf LVB-internen Praktiken, sondern berücksichtigen Best Practices verschiedener deutscher Verkehrsbetriebe. Die Unterscheidung zwischen Haltestelle und Halt, die Position des Grundes am Ende oder die Regeln zur Aufzählung sind keine LVB-spezifischen Eigenheiten, sondern finden sich in ähnlicher Form bei anderen Betreibern. Dies schafft die Grundlage für ein allgemeingültiges Format, das sich mit minimalen Anpassungen auf andere Betreiber übertragen lässt. Die Entwicklung übertragbarer Standards in der technischen Kommunikation entspricht etablierten Ansätzen zur Qualitätssicherung und Prozessoptimierung in organisationsübergreifenden Kontexten [70], [77].

4 Modellauswahl: leo-mistral-hessianai-7b

4.1 Anforderungen an das Modell

Die Auswahl eines geeigneten Sprachmodells fuer die Transformation von LVB-Verkehrsanweisungen erfordert die Beruecksichtigung mehrerer spezifischer Anforderungen. Als primaere Voraussetzung muss das Modell ueber umfassende deutschsprachige Kompetenz verfuegen, da die Verkehrsanweisungen der Leipziger Verkehrsbetriebe ausschliesslich in deutscher Sprache vorliegen. Parallel dazu muss die Modellgroesse moderate Dimensionen aufweisen, um eine lokale Ausfuehrung auf verfuegbarer Hardware zu ermoeglichen. Die Open-Source-Eigenschaft stellt eine weitere zentrale Anforderung dar, da sie sowohl DSGVO-Konformitaet gewaehrleistet als auch die Unabhaengigkeit von externen API-Diensten sicherstellt. Zudem muss das Modell Fine-Tuning-faehig sein, um eine Anpassung an die spezifische Domaene der Verkehrsanweisungen zu ermoeglichen.

Die Ausschlusskriterien fuer potenzielle Modelle ergeben sich aus den Limitierungen verkehrsspezifischer Sprachmodelle, wie sie in Abschnitt 2.1.3 diskutiert wurden. Modelle wie TrafficSafetyGPT, TransGPT und aehnliche spezialisierte Systeme scheiden aus mehreren Gruenden aus. Zum einen weisen diese Modelle eine erhebliche Sprachbarriere auf, da sie primaer auf Englisch oder Chinesisch trainiert wurden und keine deutschen Varianten existieren. Zum anderen besteht ein fundamentaler Aufgaben-Mismatch, da diese Modelle fuer Sicherheitsanalysen, Vorhersagen oder Verkehrssimulationen konzipiert wurden, nicht jedoch fuer Text-Transformationen. Hinzu kommt, dass viele dieser Modelle proprietaer sind oder auf Closed-Source-Basismodellen wie LLaMA oder ChatGLM2 aufbauen, was ihre Anwendbarkeit einschraenkt. Schliesslich sind diese Modelle nicht fuer offline-faehige lokale Ausfuehrung konzipiert worden.

Die Notwendigkeit eines deutschsprachigen Basismodells begruendet sich durch mehrere wissenschaftliche Erkenntnisse zum Cross-Language Transfer. Studien zeigen, dass die Transferleistung zwischen Sprachen stark von der linguistischen Aehnlichkeit der Sprachen abhaengt und dass bei der Anwendung englischsprachiger Modelle auf deutsche Texte erhebliche Leistungseinbussen auftreten koennen [87]. Die Fehleranfaelligkeit bei der Uebertragung von Faehigkeiten zwischen Sprachen steigt mit zunehmender linguistischer Distanz deutlich an. Die Arbeit von Ostendorff und Rehm demonstriert, dass spezialisierte Cross-Lingual Transfer-Methoden notwendig sind, um effiziente Sprachmodelle fuer ressourcenarme Sprachen zu entwickeln, da englischsprachige Modelle die spezifischen morphologischen und syntaktischen Eigenschaften des Deutschen nicht ausreichend erfassen [88]. Diese Faktoren machen eigenstaendiges Fine-Tuning auf deutschen Daten unumgaenglich.

Trotz der vergleichsweise geringen Groesse des verfuegbaren Datensatzes wird eine signifikante Verbesserung der Modellleistung erwartet. Der Anwendungs-

fall der LVB-Verkehrsanweisungen ist sehr spezifisch und eng umrissen, was den Datenbedarf reduziert. Transfer Learning von einem deutschen Basismodell minimiert den Bedarf an grossen Trainingsmengen erheblich, da das Modell bereits ueber fundamentale Sprachkenntnisse verfuegt [89]. Empirische Studien belegen, dass bei domaenenspezifischen Aufgaben auch kleine, hochwertige Datensaetze zu substanziellen Verbesserungen fuehren koennen, insbesondere wenn Parameter-Efficient Fine-Tuning Methoden zum Einsatz kommen [90], [91].

4.2 Modellvergleich und Auswahlprozess

Fuer die finale Modellauswahl wurden drei Kandidatenmodelle systematisch verglichen, die jeweils unterschiedliche Vor- und Nachteile fuer den vorliegenden Anwendungsfall aufweisen. Das erste Modell, Mistral-7B Base, stellt ein englischsprachiges Basismodell dar, das mit 7,3 Milliarden Parametern eine kompakte, aber leistungsstarke Architektur bietet. Mistral-7B nutzt fortschrittliche Mechanismen wie Grouped-Query Attention (GQA) und Sliding Window Attention (SWA), die zu einer signifikanten Beschleunigung der Inferenzgeschwindigkeit fuehren und gleichzeitig laengere Sequenzen mit reduzierten Rechenkosten verarbeiten koennen [9]. Das Modell uebertrifft Llama 2 13B ueber alle evaluierten Benchmarks hinweg und erreicht vergleichbare Leistung mit dem deutlich groesseren Llama 34B Modell. Die Baseline-Performance dieses Modells ist ausserordentlich hoch, und es wurde explizit fuer Fine-Tuning-Aufgaben optimiert. Der entscheidende Nachteil liegt jedoch darin, dass es nicht auf deutschen Texten vortrainiert wurde, was einen deutlich hoeheren Bedarf an deutschen Trainingsdaten fuer ein effektives Fine-Tuning nach sich ziehen wuerde.

Das zweite Kandidatenmodell, Mistral-7B-Instruct, repraesentiert eine instruktionsoptimierte Variante des Basismodells. Diese Version wurde bereits einem Supervised Fine-Tuning unterzogen und kann Anweisungen ohne zusaetzliches Training befolgen. Waehrend dies fuer viele Anwendungsfaelle vorteilhaft ist, entstehen beim erneuten Fine-Tuning potenzielle Konflikte. Das vorhandene Instruction Tuning koennte durch domaenenspezifisches Fine-Tuning teilweise ueberschrieben werden, was zum Phaenomen des Catastrophic Forgetting fuehren kann [92]. Empirische Studien zeigen, dass Instruct-Modelle bei erneutem Fine-Tuning zwischen 20 und 50 Prozent ihrer urspruenglichen Instruktions-Folge-Faehigkeiten einbuessen koennen, abhaengig vom verwendeten Datensatz und der Trainingsmethode [93]. Zudem ist das Modell ebenfalls nicht auf deutschen Texten vortrainiert, was die gleichen Herausforderungen wie beim Basismodell mit sich bringt.

Das dritte und letztendlich ausgewaehlte Modell, leo-mistral-hessianai-7b, kombiniert die Staerken der Mistral-7B-Architektur mit umfassender deutscher Sprachkompetenz. Dieses Modell wurde durch Continued Pretraining auf einem grossen Korpus deutscher Texte erstellt, wobei die Mistral-7B-Architektur als Ausgangspunkt diente [94]. Das LeoLM-Projekt, entwickelt in Zusammenarbeit zwischen

LAION und HessianAI, erweitert die Faehigkeiten von Mistral-7B durch ein fortgesetztes Training auf 65 Milliarden Token deliberat gefiltertem und dedupliziertem deutschem Webtext aus dem OSCAR-2301-Korpus. Evaluationen zeigen, dass LeoLM auf deutschen Benchmarks signifikant besser abschneidet als das urspruengliche Llama-2-Modell, waehrend die Leistung auf englischen Aufgaben nur geringfuegig abnimmt. Diese Charakteristik demonstriert, dass das Modell erfolgreich neue sprachliche Faehigkeiten erworben hat, ohne seine urspruenglichen Kompetenzen zu verlieren [94].

Die Wahl fiel auf leo-mistral-hessianai-7b aus mehreren strategischen Gruenden. Die deutsche Sprachkompetenz ist fuer den LVB-Anwendungsfall essentiell und reduziert den Bedarf an umfangreichen Trainingsdaten erheblich. Als Base-Variante ohne vorheriges Instruction Tuning bietet das Modell maximale Flexibilitaet beim domaenenspezifischen Fine-Tuning, ohne Gefahr von Konflikten mit bereits gelernten Instruktionsmustern. Die Kombination aus einem kleinen Datensatz und einem Base Model erweist sich als ueberlegen gegenueber dem Ansatz, ein Instruct-Model durch Prompting zu nutzen, da letzteres keine dauerhafte Anpassung an die spezifische Domaene ermoeglicht [95]. Das eigene Fine-Tuning gewaehrt zudem bessere Kontrolle ueber das Output-Format, was fuer die strukturierte Transformation von Verkehrsanweisungen von zentraler Bedeutung ist. Schliesslich ist das Modell mit 7 Milliarden Parametern hardware-kompatibel und kann auf moderater GPU-Infrastruktur ausgefuehrt werden.

4.3 Modellvarianten fuer Deployment

Nach Abschluss des Fine-Tuning-Prozesses werden zwei quantisierte Varianten des Modells erstellt, um verschiedene Deployment-Szenarien zu unterstuetzen. Die Quantisierung stellt eine etablierte Technik zur Modellkompression dar, bei der die numerische Praezision der Gewichte reduziert wird. Waehrend urspruengliche Modelle typischerweise in 32-Bit oder 16-Bit Floating-Point-Praezision vorliegen, ermoeglicht die Quantisierung eine Reduktion auf 8-Bit oder 4-Bit Integer-Formate [96].

Die erste Variante nutzt INT8-Quantisierung, die den Speicherbedarf des Modells auf etwa 7 GB VRAM reduziert. Diese Quantisierungsmethode hat sich als besonders effektiv erwiesen, da sie den Speicherbedarf halbiert, waehrend die Modellqualitaet weitgehend erhalten bleibt. Empirische Studien zeigen, dass INT8-Quantisierung die Performance auf den meisten Benchmarks typischerweise innerhalb von einem Prozent des urspruenglichen FP16-Modells haelt [97], [98]. Die Inferenzgeschwindigkeit erhoeht sich durch die reduzierte Speicherbandbreite und die Nutzung optimierter Integer-Operationen um das Zwei- bis Vierfache gegenueber dem nicht-quantisierten Modell [99].

Die zweite Variante verwendet INT4-Quantisierung, wodurch der VRAM-Bedarf

auf circa 3,5 GB sinkt. Diese aggressive Quantisierung reduziert den Speicherbedarf um 75 Prozent gegenueber FP16-Modellen. Die Qualitaetseinbusse faellt bei INT4-Quantisierung groesser aus als bei INT8, bewegt sich jedoch fuer die meisten Anwendungen in akzeptablem Rahmen. Untersuchungen zeigen, dass INT4-quantisierte Modelle bei verschiedenen Aufgaben Leistungseinbussen von 2 bis 5 Prozent aufweisen koennen, wobei spezialisierte Quantisierungsmethoden wie Activation-Aware Weight Quantization (AWQ) diese Degradation weiter minimieren koennen [98], [100]. Die Inferenzgeschwindigkeit kann sich durch INT4-Quantisierung bis zu achtfach verbessern, insbesondere in latenz-orientierten Szenarien [100].

Die Entscheidung, ausschliesslich quantisierte Varianten fuer die finale Evaluation zu verwenden, basiert auf Beobachtungen aus initialen Tests. Diese zeigten unerwarteterweise, dass quantisierte Modelle in bestimmten Metriken das nichtquantisierte Modell uebertreffen konnten. Dieses Phaenomen wird auf die Regularisierungseffekte der Quantisierung zurueckgefuehrt, die in manchen Faellen zu einer verbesserten Generalisierung fuehren koennen. Fuer den finalen Vergleich in Kapitel 7 werden daher beide quantisierten Varianten evaluiert, um die optimale Balance zwischen Ressourceneffizienz und Modellqualitaet zu bestimmen.

Die erwarteten Trade-offs zwischen den Varianten folgen etablierten Mustern der Modellkompression. Waehrend INT4 maximale Speichereffizienz und Inferenzgeschwindigkeit bietet, geht dies mit dem hoechsten Risiko fuer Qualitaetsverluste einher. INT8 repraesentiert einen ausgewogenen Kompromiss, der substanzielle Effizienzgewinne mit minimaler Qualitaetsdegradation verbindet. Die theoretischen Grundlagen dieser Trade-offs wurden in Abschnitt 2.3.2 dargelegt. Die praktische Evaluation in Kapitel 7 wird zeigen, welche Quantisierungsstufe sich fuer die spezifische Aufgabe der Verkehrsanweisungstransformation als optimal erweist.

5 Datensatzerstellung und -aufbereitung

5.1 Datenerhebung

Die Grundlage für die Erstellung des Trainingsdatensatzes bilden technische Verkehrsanweisungen der Leipziger Verkehrsbetriebe, die über einen Zeitraum von mehreren Jahren im operativen Betrieb erstellt wurden. Diese Verkehrsanweisungen dokumentieren verschiedene Betriebsänderungen, Baumaßnahmen und Umleitungen und dienen primär der internen Koordination zwischen verschiedenen Abteilungen. Die Sammlung umfasst sowohl aktuelle als auch archivierte Anweisungen, um eine breite Abdeckung verschiedener Verkehrssituationen zu gewährleisten.

Die Auswahl der Verkehrsanweisungen für die Datensatzerstellung erfolgte anhand mehrerer Kriterien. Zunächst wurden Anweisungen priorisiert, die bereits manuell erstellte Fahrgastinformationen aufweisen, da diese als Referenz für die korrekte Transformation dienen. Zusätzlich wurden Anweisungen ausgewählt, die verschiedene Komplexitätsdimensionen gemäß Abschnitt 3.1.1 abdecken, um eine ausgewogene Repräsentation unterschiedlicher Verkehrssituationen zu gewährleisten. Die Aktualität der Anweisungen spielte ebenfalls eine Rolle, wobei sowohl zeitnahe als auch historische Beispiele einbezogen wurden, um die Generalisierungsfähigkeit des Modells zu fördern.

Die Rohdaten liegen in verschiedenen Formaten vor, primär als PDF-Dokumente und teilweise als strukturierte Textdateien. Die technischen Verkehrsanweisungen folgen der in Abschnitt 3.1.2 beschriebenen Standardstruktur, weisen jedoch Variationen in Detailgrad und Formulierung auf. Diese Heterogenität im Rohdatenbestand erfordert eine systematische Aufbereitung und Annotation, die in den folgenden Abschnitten beschrieben wird.

Es ist wichtig zu betonen, dass die in Abschnitt 3.1.3 analysierten 185 Beispiele von Fahrgastinformationen verschiedener deutscher Verkehrsbetriebe nicht Teil des Trainingsdatensatzes sind. Diese Beispiele stammen aus einer separaten Analyse-Datenbank und dienten ausschließlich der Entwicklung des Regelwerks zur Formulierung von Fahrgastinformationen. Die Trainingsbeispiele wurden hingegen vollständig neu erstellt, indem technische Verkehrsanweisungen der LVB unter Anwendung der entwickelten Regeln manuell in Fahrgastinformationen transformiert wurden. Diese klare Trennung zwischen Regelentwicklung und Datensatzerstellung gewährleistet, dass der Trainingsdatensatz spezifisch auf die LVB-Anforderungen zugeschnitten ist und nicht durch externe Formulierungskonventionen verzerrt wird.

Die Charakterisierung des Rohdatenbestands zeigt eine natürliche Verteilung verschiedener Maßnahmentypen. Sperrungen und Umleitungen bilden die häufigsten Kategorien, gefolgt von Ersatzverkehren und Verkürzungen. Halteausfälle und Komplettausfälle von Linien treten seltener auf, sind jedoch für eine vollständige

Abdeckung der Domäne essentiell. Die zeitliche Verteilung der Verkehrsanweisungen umfasst sowohl kurzfristige Baumaßnahmen von wenigen Tagen als auch langfristige Projekte über mehrere Monate. Diese Diversität im Rohdatenbestand bildet die Grundlage für einen ausgewogenen Trainingsdatensatz, der die praktischen Anforderungen im operativen Betrieb der Leipziger Verkehrsbetriebe widerspiegelt.

5.2 Datensatzentwicklung für Fine-Tuning

Die Entwicklung eines spezialisierten Trainingsdatensatzes bildet die Grundlage für das Fine-Tuning des Sprachmodells. Dieses Kapitel beschreibt die methodische Vorgehensweise bei der Erstellung der Input-Output-Paare, die Strukturierung des Datensatzes sowie die angewendeten Strategien zur Optimierung der Trainingsqualität trotz begrenzter Datenmenge. Die praktische Umsetzung orientiert sich dabei an den in Kapitel 2 dargelegten theoretischen Grundlagen des Transfer Learning und der parameter-effizienten Fine-Tuning-Methoden.

5.2.1 Annotationsstrategie

Die Transformation technischer Verkehrsanweisungen in nutzergerechte Fahrgastinformationen erfordert ein systematisches Regelwerk, das die sprachlichen und strukturellen Anforderungen der Zielformate abbildet. Die in Abschnitt 3.1.2 identifizierten Konventionen wurden hierzu in operationalisierbare Transformationsregeln überführt, die als Grundlage für die manuelle Annotation der Trainingsdaten dienen. Diese Annotationsstrategie gewährleistet die Konsistenz und Qualität der Input-Output-Paare, die für das Fine-Tuning des Sprachmodells verwendet werden.

Das entwickelte Regelwerk gliedert sich in drei Hauptkategorien, die verschiedene Aspekte der Texttransformation adressieren. Strukturregeln definieren den grundlegenden Aufbau der Fahrgastinformationen, insbesondere die Reihenfolge der Informationselemente sowie Ausnahmen für Sonderfälle wie Events oder Endmeldungen. Die Grundstruktur folgt einem konsistenten Schema, bei dem zunächst der Gültigkeitszeitraum genannt wird, gefolgt von den betroffenen Linien und den konkreten Maßnahmen. Den Abschluss bildet die Begründung für die Änderung. Diese Struktur priorisiert die für die unmittelbare Reiseplanung relevanten Informationen, während der Grund als kontextuelle Information nachgestellt wird.

Formulierungsregeln legen fest, wie spezifische Maßnahmen sprachlich umgesetzt werden. Hierzu zählen unter anderem die Formulierung von Umleitungen, Halteausfällen oder Ersatzverkehren. Eine zentrale Transformationsregel stellt die Fahrzeugtyp-Spezifikation dar. Während technische Verkehrsanweisungen häufig die generische Formulierung wie beispielsweise ”Linie 15”ver-

wenden, erfordert die Fahrgastinformation die explizite Nennung des Fahrzeugtyps als "TRAM 15". Diese Regel erhöht die Eindeutigkeit, insbesondere in städtischen Verkehrsnetzen, in denen Bus- und Straßenbahnlinien mit identischen oder ähnlichen Nummern parallel verkehren.

Die Pluralisierungsregel optimiert die Lesbarkeit bei mehreren betroffenen Linien desselben Fahrzeugtyps. Anstelle der redundanten Aufzählung wird der Fahrzeugtyp nur einmal vorangestellt, beispielsweise "BUS 71, 73 und 90änstelle von "BUS 71, BUS 73 und BUS 90". Diese Regel reduziert nicht nur die Zeichenzahl, sondern verbessert auch die Scannbarkeit der Information für Fahrgäste, die schnell erfassen müssen, ob ihre Linie betroffen ist.

Formatierungsregeln betreffen Details wie die Ausschreibung von Abkürzungen, die Verwendung von Anführungszeichen bei Haltestellennamen oder die Pluralisierung bei mehreren Linien desselben Fahrzeugtyps. Die Ausschreibung fachspezifischer Abkürzungen trägt zur Verständlichkeit für alle Fahrgäste bei. Während Abkürzungen wie "Hst."für Haltestelle, SStrbhf."für Straßenbahnhof oder "örtl."für örtlicher in technischen Dokumenten üblich sind, werden sie in Fahrgastinformationen konsequent ausgeschrieben. Allgemein bekannte Straßenabkürzungen wie str."bleiben hingegen erhalten, da sie auch außerhalb des Verkehrskontextes gebräuchlich sind.

Für Ansagetexte gelten modifizierte Regeln, die der auditiven Rezeption und der begrenzten Aufmerksamkeitsspanne während der Fahrt Rechnung tragen. Die Struktur beginnt stets mit der Begrüßung SSehr geehrte Fahrgäste", gefolgt von der unmittelbar integrierten Begründung "wegen [Grund]ünd den Maßnahmen. Zeitangaben und Liniennummern werden weggelassen, da sich die Ansage bereits an Fahrgäste in einem bestimmten Fahrzeug richtet. Der Fahrzeugbezug erfolgt über "dieser Busöder "diese Straßenbahn". Diese Kompaktheit ist notwendig, da Ansagetexte in kurzer Zeit verständlich sein müssen und die kognitive Verarbeitung auditiver Informationen eine Reduktion auf die unmittelbar relevanten Fakten erfordert.

Die Entwicklung der Satzbausteine erfolgte durch systematische Kategorisierung nach ihrer Funktion innerhalb der Fahrgastinformation. Einleitungselemente umfassen Zeitangaben und Event-Anlässe. Problemdefinitionen beschreiben die Art der Beeinträchtigung, beispielsweise Verkürzungen, Umleitungen oder Halteausfälle. Lösungselemente kommunizieren verfügbare Alternativen wie Ersatzverkehre oder Ersatzhaltestellen. Begründungen erläutern den Anlass der Maßnahme. Diese funktionale Kategorisierung ermöglicht eine flexible Kombination der Bausteine je nach konkreter Verkehrssituation, wobei die Grundstruktur gewahrt bleibt.

Für verschiedene Maßnahmentypen wurden spezifische Templates entwickelt, die als strukturelles Grundgerüst dienen. Ein Template für Verkürzungen lautet beispielsweise: "[Linie mit Fahrzeugtyp] fährt verkürzt bis und ab [Endstelle]". Für Umleitungen wird das Template "[Linie mit Fahrzeugtyp] fährt mit Um-

leitung über [Straße1], [Straße2] und [Straße3]"verwendet. Bei Halteausfällen kommt das Template "Die [Haltestelle/Halt] '[Name]' entfällt ersatzlos. Grund dafür ist [Beschreibung]ßum Einsatz. Diese Templates werden im konkreten Anwendungsfall mit den spezifischen Informationen aus der Verkehrsanweisung gefüllt.

Die Validierung der Regeln erfolgte iterativ durch Abgleich mit bestehenden Fahrgastinformationen auf der Plattform him.l.de sowie durch Rückmeldungen der LVB-Mitarbeitenden. Dabei wurden die Regeln anhand realer Verkehrsanweisungen getestet und bei Bedarf präzisiert, um eine konsistente und praxistaugliche Anwendung sicherzustellen. Die resultierenden Regeln bilden die Grundlage für die manuelle Erstellung der Trainingsbeispiele, deren Qualität durch die in Abschnitt 5.2.2 beschriebenen automatisierten Validierungsprozesse sichergestellt wird.

5.2.2 Qualitätssicherung und Validierung

Um die Qualität und Konsistenz des Trainingsdatensatzes sicherzustellen, wurde eine mehrstufige automatisierte Validierung mittels Python-Skripten durchgeführt. Diese Validierungsprozesse prüfen strukturelle, inhaltliche und semantische Aspekte der annotierten Input-Output-Paare und identifizieren potenzielle Inkonsistenzen, die die Qualität des Fine-Tunings beeinträchtigen könnten. Die Implementierung der Validierungslogik erfolgte in modularer Form, sodass einzelne Prüfungen sowohl isoliert als auch als Gesamtprozess ausgeführt werden können.

Strukturvalidierung Die Strukturvalidierung prüft die Einhaltung der in Abschnitt 3.1.2 definierten Strukturregeln für Fahrgastinformationen und Ansagetexte. Für Fahrgastinformationen wird überprüft, ob die Zeitangaben am Anfang der Meldung stehen und die Begründung mit der Formulierung "Grund dafüräm Ende positioniert ist. Bei Ansagetexten erfolgt die Validierung der einleitenden Begrüßungsformel SSehr geehrte Fahrgästeäm Textbeginn.

Die Strukturvalidierung ergab für beide Datentypen eine vollständige Regelkonformität. Alle 450 Einträge der Fahrgastinformationen wiesen die korrekte Positionierung der Zeitangaben am Anfang sowie der Begründung am Ende auf. Ebenso begannen alle 350 Ansagetexte mit der vorgeschriebenen Begrüßungsformel. Diese konsistente Struktureinhaltung bildet die Grundlage für das spätere Lernen der strukturellen Muster durch das Sprachmodell.

Linien- und Fahrzeugtyp-Validierung Die Linienvalidierung gleicht alle im Datensatz genannten Liniennummern mit dem aktuellen Linienverzeichnis der Leipziger Verkehrsbetriebe ab. Zusätzlich wird die korrekte Zuordnung der Fahrzeugtypen zu den Liniennummern überprüft, da die eindeutige Spezifikation

als ”TRAMöder ”BUSëine zentrale Anforderung der Fahrgastinformation darstellt. Diese Validierung verhindert das Training auf veralteten oder ungültigen Linienbezeichnungen sowie auf fehlerhaften Zuordnungen zwischen Liniennummer und Fahrzeugtyp.

Von den 450 Fahrgastinformationen enthielten 434 Einträge Linienbezeichnungen, was insgesamt 658 Linienerwähnungen entspricht. Die Validierung bestätigte die Gültigkeit aller 658 Linienerwähnungen, sodass keine veralteten oder fehlerhaften Linienbezeichnungen im Trainingsdatensatz vorhanden sind. Diese vollständige Validität ist essentiell, da fehlerhafte Linienbezeichnungen zu falschen Ausgaben in der späteren Modellanwendung führen würden.

Entitäten-Konsistenzprüfung Die Entitäten-Konsistenzprüfung extrahiert Named Entities wie Haltestellennamen und Straßenbezeichnungen aus Input und Output und gleicht diese ab. Das Ziel besteht darin, Informationsverluste durch fehlende Entitäten im Output sowie Halluzinationen durch zusätzliche, im Input nicht genannte Entitäten zu identifizieren. Die Prüfung berücksichtigt, dass Inputs mehr Informationen als Outputs enthalten dürfen, da nicht alle technischen Details für Fahrgäste relevant sind. Outputs dürfen jedoch keine Entitäten enthalten, die nicht im Input vorkommen.

Die Konsistenzprüfung ergab für 419 der 450 Fahrgastinformationen eine vollständige Übereinstimmung der Entitäten zwischen Input und Output, was einer Quote von 93,1 Prozent entspricht. Bei 31 Einträgen wurden mögliche Inkonsistenzen identifiziert, die in einer manuellen Nachprüfung evaluiert wurden. Diese Inkonsistenzen betrafen überwiegend unterschiedliche Schreibweisen von Straßennamen sowie Fälle, in denen im Output zusätzliche Kontextinformationen gegeben wurden, die nicht explizit im strukturierten Input aufgeführt waren, jedoch aus der ursprünglichen Verkehrsanweisung stammten.

Maßnahmentyp-Validierung Die Maßnahmentyp-Validierung erkennt automatisch die Art der verkehrlichen Maßnahme sowohl im Input als auch im Output. Unterschieden werden dabei die Maßnahmentypen Sperrung, Umleitung, Verkürzung, Ersatzverkehr, Halteausfall sowie Komplettausfall einer Linie. Die Validierung prüft, ob die Maßnahmeninformation korrekt vom Input in den Output übertragen wurde, und erstellt eine Verteilungsstatistik über alle Maßnahmentypen im Datensatz.

Bei den Fahrgastinformationen zeigt sich eine charakteristische Transformation zwischen Input und Output. Während im Input Sperrungen mit 343 Erwähnungen (76,2 Prozent) den häufigsten Maßnahmentyp darstellen, sind im Output Umleitungen mit 335 Erwähnungen (74,4 Prozent) dominant. Diese Verschiebung ist erwünscht und entspricht der fahrgastorientierten Kommunikation, die nicht die technische Ursache, sondern die resultierende Maßnahme in den Vordergrund stellt. Eine Sperrung führt häufig zu einer Umleitung, weshalb die Umleitung

für Fahrgäste die relevante Information darstellt.

Die Konsistenz der Maßnahmenübertragung liegt bei 79,6 Prozent für Fahrgastinformationen und 85,7 Prozent für Ansagetexte. Diese Werte reflektieren die bewusste Transformation von technischen Ursachenbeschreibungen in handlungsorientierte Maßnahmenbeschreibungen. In 92 Fällen bei Fahrgastinformationen und 50 Fällen bei Ansagetexten wurden Diskrepanzen zwischen Input- und Output-Maßnahmen identifiziert. Die manuelle Überprüfung dieser Fälle bestätigte, dass die meisten Diskrepanzen auf legitime Transformationen zurückzuführen sind, bei denen beispielsweise eine Sperrung im Input korrekt zu einer Umleitung im Output führt.

Die Verteilung der Maßnahmentypen im Datensatz zeigt eine ausgewogene Abdeckung verschiedener Verkehrssituationen. Neben den häufigen Umleitungen und Sperrungen sind auch Ersatzverkehre mit 186 Erwähnungen (41,3 Prozent im Input) sowie Verkürzungen mit 140 Erwähnungen (31,1 Prozent im Input) substantiell vertreten. Halteausfälle mit 66 Erwähnungen (14,7 Prozent) und Komplettausfälle mit 10 Erwähnungen (2,2 Prozent) bilden die selteneren, aber dennoch wichtigen Kategorien ab.

Richtungsspezifische Angaben-Validierung Die Validierung richtungsspezifischer Angaben prüft die korrekte Formatierung von Richtungsinformationen, die bei vielen Verkehrsmaßnahmen nur eine Fahrtrichtung betreffen. Das erwartete Format folgt dem Schema "Richtung [Zielort]:", wobei die Einhaltung der Reihenfolge sowie die Verwendung des Doppelpunkts überprüft werden. Zusätzlich wird die Konsistenz bei mehreren Richtungsangaben im selben Text validiert.

Von den 450 Fahrgastinformationen enthalten 350 Einträge (77,8 Prozent) Richtungsangaben, wobei 204 Einträge (45,3 Prozent) mehrere Richtungen adressieren. Die Verteilung der Richtungsanzahl zeigt, dass die meisten Meldungen eine oder zwei Richtungen behandeln: 146 Einträge enthalten eine Richtung (32,4 Prozent) und 118 Einträge zwei Richtungen (26,2 Prozent). Komplexere Fälle mit drei bis elf Richtungen sind mit insgesamt 86 Einträgen (19,1 Prozent) seltener, bilden jedoch wichtige Szenarien ab, in denen mehrere Linien mit unterschiedlichen Richtungsverläufen betroffen sind.

Bei Ansagetexten liegt der Anteil richtungsspezifischer Angaben mit 183 von 350 Einträgen (52,3 Prozent) niedriger. Dies entspricht dem Kommunikationsziel von Ansagetexten, die sich an bereits im Fahrzeug befindliche Fahrgäste richten und daher häufig auf die konkrete Fahrtrichtung fokussieren. Der Anteil von Einträgen mit mehreren Richtungen beträgt bei Ansagetexten 92 (26,3 Prozent), was die stärkere Fokussierung auf einzelne Richtungen unterstreicht.

Zusätzliche Validierungsprüfungen Ergänzend zu den inhaltlichen Validierungen wurden technische Prüfungen implementiert, die die Datenintegrität sicherstellen. Die Duplikatsprüfung identifiziert identische oder nahezu identische Trainingsbeispiele, die zu einer ineffizienten Nutzung der Trainingsressourcen führen würden. Die Zeichensatzvalidierung stellt die korrekte UTF-8-Kodierung sowie die fehlerfreie Verarbeitung von Sonderzeichen und Umlauten sicher. Die Längenvalidierung überprüft, dass alle Input- und Output-Sequenzen innerhalb der zulässigen Token-Grenzen liegen. Die Vollständigkeitsprüfung gewährleistet, dass alle Pflichtfelder der Trainingsbeispiele vorhanden sind, und die Format-Konsistenzprüfung validiert die einheitliche Struktur aller Einträge gemäß dem definierten JSON-Schema.

Iterativer Korrekturprozess Die identifizierten Inkonsistenzen wurden in einem iterativen Prozess manuell nachbearbeitet. Dabei wurden die 31 Entitäten-Inkonsistenzen sowie die 92 Maßnahmentyp-Diskrepanzen bei Fahrgastinformationen überprüft und gegebenenfalls korrigiert. In vielen Fällen handelte es sich nicht um tatsächliche Fehler, sondern um legitime Transformationen, die von den automatischen Prüfungen als potenzielle Inkonsistenzen markiert wurden. Die finale Validierung nach dem Korrekturprozess bestätigte die Datenqualität für das Fine-Tuning.

Die mehrstufige Validierung verhindert das Training auf fehlerhaften Daten, reduziert Halluzinationen bei Ortsnamen und Linienbezeichnungen und sichert die Konsistenz mit den LVB-Vorgaben. Das systematische Validierungsverfahren erhöht das Vertrauen in die Trainingsdaten und bildet eine wichtige Qualitätssicherungsmaßnahme im Gesamtprozess der Datensatzerstellung.

5.2.3 Datenaugmentierung

Die Datenaugmentierung für den Trainingsdatensatz basiert auf den in Abschnitt 2.2.3 vorgestellten theoretischen Grundlagen zu Data Augmentation und Balanced Datasets. Die implementierten Strategien zielen darauf ab, einen ausgewogenen und qualitativ hochwertigen Datensatz zu schaffen, der trotz begrenzter Ausgangsdaten eine breite Abdeckung verschiedener Verkehrssituationen ermöglicht. Die Augmentierungsstrategien wurden so konzipiert, dass sie die Realitätstreue der Daten wahren und gleichzeitig die Generalisierungsfähigkeit des Modells fördern.

Ein zentrales Element der Augmentierungsstrategie bestand darin, aus einer einzelnen Verkehrsanweisung nicht nur die erwarteten Standardformate als Trainingsbeispiele zu generieren, sondern darüber hinaus weitere Varianten zu erstellen. Insbesondere wurden verschiedene Kombinationen mehrerer Linien aus einer Anweisung abgeleitet, um den Datensatz ausgeglichener zu gestalten und zu verhindern, dass das Modell sich auf die Extraktion von nur einer Linie pro

Anweisung spezialisiert. Da die Mehrheit der Rohdaten Einzellinien-Meldungen umfasst, wurde durch diese gezielte Erweiterung eine stärkere Repräsentation komplexerer Szenarien erreicht. Bei der Generierung dieser zusätzlichen Beispiele wurde konsequent darauf geachtet, dass sämtliche zuvor entwickelten Regeln zur Erstellung und Formatierung der Trainingsbeispiele eingehalten werden. Dadurch ist sichergestellt, dass die synthetisch erzeugten Beispiele die gleiche Qualität und Konsistenz wie die natürlichen, manuell annotierten Daten aufweisen.

Datenvariationen nach Komplexitätskategorien Der Trainingsdatensatz wurde bewusst so zusammengestellt, dass er alle in Abschnitt 3.1.1 definierten Komplexitätsdimensionen von Verkehrsmeldungen abdeckt. Diese Dimensionen umfassen die Anzahl betroffener Linien, die Richtungsspezifität der Maßnahmen, zeitliche Staffelungen, die Art der verkehrlichen Maßnahmen sowie räumliche Ausdehnung und Interdependenzen zwischen verschiedenen Maßnahmen. Die Verteilung dieser Dimensionen im Datensatz orientiert sich an der empirischen Häufigkeit in den Produktivdaten der Leipziger Verkehrsbetriebe, wobei gezielt unterrepräsentierte Komplexitätsgrade durch Augmentierung verstärkt wurden.

Verkehrsanweisungen mit geringerer Komplexität, die beispielsweise nur eine einzelne Linie in einer Fahrtrichtung betreffen, bilden die Grundlage des Datensatzes und sind entsprechend ihrer Häufigkeit im operativen Betrieb stärker vertreten. Verkehrsanweisungen mit höherer Komplexität, die mehrere Linien mit unterschiedlichen Maßnahmen oder mehrere Zeiträume kombinieren, sind in geringerem Maße vertreten, jedoch in ausreichender Anzahl vorhanden, um eine Generalisierung auf diese anspruchsvolleren Fälle zu ermöglichen. Verkehrsanweisungen mittlerer Komplexität, bei denen mehrere Linien identische Maßnahmen erfahren, nehmen eine wichtige Rolle ein, da sie die effiziente Darstellung gebündelter Informationen trainieren, die für die praktische Anwendung relevant ist.

Die Verteilungsstrategie folgt dem Prinzip des Balanced Sampling für kleine Datensätze und vermeidet extreme Imbalancen zwischen den Komplexitätsgraden. Das Verhältnis zwischen den am häufigsten und am seltensten vertretenen Komplexitätsdimensionen wurde auf maximal 1:3 begrenzt, um eine ausgewogene Lernverteilung zu gewährleisten. Diese Balance zwischen natürlicher Häufigkeit und künstlicher Augmentierung seltener Fälle entspricht den Empfehlungen der Fachliteratur zur Vermeidung von Class Imbalance bei der Modellperformance.

Disaggregation gebündelter Verkehrsanweisungen Eine zentrale Augmentierungsstrategie besteht in der systematischen Disaggregation gebündelter Verkehrsanweisungen. Aus einer Quell-Verkehrsanweisung wurden nicht nur die standardmäßig vorgesehenen Texte erzeugt, sondern gebündelte Meldungen zusätzlich in separate Einzelansagen aufgeteilt. Verkehrsanweisungen, die mehrere Linien oder richtungsspezifische Maßnahmen in einem Text zusammenfassen, wurden

sowohl in der gebündelten Form als auch als separate Einzelmeldungen in den Datensatz aufgenommen.

Das systematische Vorgehen durchläuft alle Komplexitätsdimensionen, wie sie in Abschnitt 3.1.1 definiert wurden. Meldungen mit höherer Komplexität, die gebündelte Informationen zu mehreren Linien oder Richtungen enthalten, wurden in ihre Bestandteile zerlegt und als separate Einzelmeldungen ergänzt. Eine Meldung der Form "Linien 1 und 2 werden umgeleitet"wird beispielsweise zusätzlich als "Linie 1 wird umgeleitetünd "Linie 2 wird umgeleitetïn den Datensatz aufgenommen. Diese Strategie verfolgt das Ziel einer ausgewogenen Verteilung über alle Komplexitätsstufen und entspricht dem Konzept der Balanced Datasets, das nachweislich die Generalisierung verbessert.

Die Disaggregation ermöglicht es dem Modell, sowohl kompakte gebündelte Darstellungen als auch detaillierte Einzeldarstellungen zu erlernen. In der späteren Anwendung kann das Modell flexibel zwischen beiden Darstellungsformen wählen, abhängig vom Kontext der Anfrage. Diese Flexibilität ist für die praktische Nutzung essentiell, da sowohl Szenarien existieren, in denen gebündelte Informationen bevorzugt werden, als auch solche, in denen separate Einzelmeldungen angemessener sind.

Synthetische Beispiele mit realen Entitäten Für unterrepräsentierte Kategorien wurden synthetische Trainingsbeispiele generiert, die auf einem Template-basierten Ansatz mit strikter Constraint-Enforcement basieren. Diese Strategie orientiert sich an den Empfehlungen zur synthetischen Datengenerierung für Natural Language Processing und gewährleistet gleichzeitig die Realitätstreue der erzeugten Beispiele.

Die synthetische Generierung unterliegt strengen Einschränkungen zur Wahrung der Realitätstreue. Es werden ausschließlich real existierende Liniennummern des LVB-Netzes verwendet, wobei die korrekte Zuordnung von Fahrzeugtypen zu Liniennummern gewährleistet wird. Alle Haltestellennamen und Straßenbezeichnungen stammen aus einer validierten Entitätsdatenbank, die auf den aktuellen Netzplänen der Leipziger Verkehrsbetriebe basiert. Die generierten Texte halten die in Abschnitt **??** definierten Strukturregeln ein, um Konsistenz mit den manuell annotierten Beispielen zu gewährleisten.

Die Verwendung einer validierten Entitätsdatenbank verhindert Halluzinationen durch nicht existierende Orte oder Linien. Jede synthetisch erzeugte Verkehrssituation ist prinzipiell im realen Betrieb umsetzbar, auch wenn sie möglicherweise noch nicht aufgetreten ist. Diese Realitätstreue unterscheidet die implementierte Augmentierungsstrategie von rein stochastischen Ansätzen, die beliebige Textvariationen erzeugen könnten. Der Anteil synthetischer Daten im domänenspezifischen Datensatz beträgt etwa 20 bis 30 Prozent, was den Empfehlungen für kleine Datensätze entspricht und eine Balance zwischen realen Daten mit höherer Validität und synthetischen Daten mit größerer Abdeckung herstellt.

Balance der Maßnahmentypen Die gleichmäßige Verteilung verschiedener Maßnahmentypen im Datensatz wurde durch gezieltes Monitoring und Augmentierung sichergestellt. Die in Abschnitt 5.2.2 beschriebene Maßnahmentyp-Validierung identifiziert die Verteilung von Sperrungen, Umleitungen, Verkürzungen, Ersatzverkehren, Halteausfällen sowie Kombinationen mehrerer Maßnahmen. Unterrepräsentierte Maßnahmentypen wurden durch synthetische Beispiele ergänzt, um Class Imbalance zu vermeiden, der nachweislich die Modellperformance beeinträchtigt.

Die finale Verteilung zeigt eine substantielle Abdeckung aller relevanten Maßnahmentypen. Umleitungen und Sperrungen bilden mit jeweils über 300 Erwähnungen die häufigsten Kategorien ab. Ersatzverkehre sind mit 186 Erwähnungen (41,3 Prozent) und Verkürzungen mit 140 Erwähnungen (31,1 Prozent) im Input der Fahrgastinformationen substantiell vertreten. Selbst die selteneren Kategorien Halteausfälle mit 66 Erwähnungen (14,7 Prozent) und Komplettausfälle mit 10 Erwähnungen (2,2 Prozent) sind in ausreichender Anzahl vorhanden, um eine Generalisierung auf diese Fälle zu ermöglichen.

Ansagetexte Ansagetexte wurden als separate Kategorie behandelt und entsprechend ihrer Häufigkeit in den Quelldaten in den Datensatz aufgenommen. Die Anzahl von 350 Ansagetexten resultiert aus der Anpassung an die neuen Strukturvorgaben sowie der Disaggregation gebündelter Anweisungen. Im Gegensatz zu Fahrgastinformationen erfolgte bei Ansagetexten keine weitere Disaggregation nach Linien, da die entscheidenden Faktoren Kompaktheit der Information und Streckenabschnittsbezogenheit dies nicht erfordern.

Der Fokus bei Ansagetexten liegt auf der korrekten Abgrenzung von Streckenabschnitten, der Priorisierung fahrgastrelevanter Informationen sowie der Vermeidung redundanter Details. Diese Anforderungen unterscheiden sich von denen der Fahrgastinformationen und rechtfertigen die separate Behandlung als eigener Datentyp. Die Verteilung der Maßnahmentypen bei Ansagetexten zeigt mit 73,7 Prozent Umleitungen im Output eine noch stärkere Fokussierung auf handlungsorientierte Informationen als bei Fahrgastinformationen.

Spezialtraining: Linien-Fahrzeugtyp-Zuordnung Zur Sicherstellung der korrekten Zuordnung von Liniennummern zu Fahrzeugtypen wurden 150 dedizierte Trainingsbeispiele erstellt, die explizit die Beziehung zwischen Liniennummer und Fahrzeugtyp trainieren. Diese Beispiele adressieren alle Linien im LVB-Netz und wurden mit dem in Abschnitt 5.2.2 beschriebenen Validierungsskript auf Korrektheit geprüft.

Die explizite Aufnahme dieser Zuordnungen als eigenständige Trainingsbeispiele verstärkt das Lernen dieser kritischen Information, die für die korrekte Ausgabe von Fahrgastinformationen essentiell ist. Die 150 Beispiele umfassen sowohl einfache Zuordnungen einzelner Linien als auch Beispiele mit mehreren Lini-

en, um verschiedene Kontexte abzudecken. Diese fokussierte Trainingskomponente ergänzt die allgemeinen Verkehrsmeldungen und stellt sicher, dass die Fahrzeugtyp-Spezifikation konsistent angewendet wird.

Verzicht auf allgemeine deutschsprachige und englische Daten Ursprünglich war die Integration allgemeiner deutschsprachiger Trainingsbeispiele aus öffentlichen NLP-Datensätzen geplant, um Catastrophic Forgetting während des domänenspezifischen Fine-Tunings zu verhindern. Ebenso wurde erwogen, englische Trainingsbeispiele zur Erhaltung der mehrsprachigen Fähigkeiten des Basismodells LeoLM einzubeziehen. Diese Strategien entsprechen dem Continual Learning Approach und sind in der Fachliteratur für domänenspezifisches Fine-Tuning empfohlen.

In der praktischen Umsetzung führte die Integration allgemeiner Daten jedoch zu erheblichen Schwierigkeiten bei der Balance zwischen Domänenspezialisierung und Erhalt allgemeiner Sprachfähigkeiten. Erste Trainingsversuche mit einem gemischten Datensatz zeigten Overfitting-Tendenzen, wobei das Modell Schwierigkeiten hatte, die spezifischen Anforderungen der Verkehrsinformations-Transformation zu erlernen, während gleichzeitig die allgemeinen Fähigkeiten erhalten blieben. Die Versuche, diesen Trade-off durch Anpassung der Mixing-Ratio zu optimieren, führten nicht zu zufriedenstellenden Ergebnissen.

Als Konsequenz wurde die Trainingsstrategie auf ausschließlich spezialisierte Daten fokussiert, wobei die Learning Rate auf 0,0001 reduziert wurde, um sanfteres Lernen zu ermöglichen. Der finale Trainingsdatensatz umfasst daher 950 rein domänenspezifische Samples ohne Integration allgemeiner deutscher oder englischer Daten. Diese Entscheidung priorisiert die Qualität der Domänenspezialisierung gegenüber dem Erhalt allgemeiner Sprachfähigkeiten, was für die fokussierte Anwendung im LVB-Kontext angemessen ist. Die Implikationen dieser Entscheidung sowie alternative Ansätze werden in Abschnitt 5.2.5 diskutiert.

Herausforderungen der Augmentierung Die Implementierung der Augmentierungsstrategien erforderte die Balance zwischen einfachen und komplexen Beispielen, die Vermeidung von Überanpassung auf häufige Muster sowie die Sicherstellung korrekter Darstellungskonventionen bei der synthetischen Generierung. Der Trade-off zwischen Domänenspezialisierung und allgemeiner Sprachfähigkeit stellte sich als zentrale Herausforderung heraus, die letztlich zur Fokussierung auf ausschließlich spezialisierte Daten führte. Diese Herausforderungen und die gewählten Lösungsansätze prägen den finalen Datensatz und beeinflussen die Trainingsstrategie, wie in den folgenden Abschnitten dargelegt wird.

5.2.4 Datensatzstruktur

Der finale Trainingsdatensatz umfasst 950 spezialisierte Trainingsbeispiele, die auf drei Datentypen verteilt sind. Die Fahrgastinformationen bilden mit 450 Samples (47,4 Prozent) die größte Kategorie und decken die vollständige Bandbreite an Verkehrssituationen ab, die in schriftlichen Fahrgastinformationen kommuniziert werden. Die Ansagetexte umfassen 350 Samples (36,8 Prozent) und adressieren die spezifischen Anforderungen auditiver Kommunikation in Fahrzeugen. Die Linien-Fahrzeugtyp-Zuordnungen stellen mit 150 Samples (15,8 Prozent) einen fokussierten Trainingsanteil dar, der die korrekte Zuordnung von Liniennummern zu Fahrzeugtypen verstärkt.

Die Aufteilung des Datensatzes in Trainings-, Validierungs- und Testdaten erfolgte stratifiziert, um in allen Splits eine repräsentative Verteilung der drei Datentypen zu gewährleisten. Für die Trainings- und Validierungsaufteilung wurde ein Verhältnis von 92:8 gewählt, was angesichts des begrenzten Gesamtumfangs des Datensatzes einen Kompromiss zwischen ausreichender Trainingsdatenmenge und aussagekräftiger Validierung darstellt. Der Trainingssplit umfasst 874 Samples, die sich auf 414 Fahrgastinformationen, 322 Ansagetexte und 138 Linien-Fahrzeugtyp-Zuordnungen verteilen. Der Validierungssplit enthält 76 Samples mit 36 Fahrgastinformationen, 28 Ansagetexten und 12 Linien-Fahrzeugtyp-Zuordnungen.

Zusätzlich wurde ein separater Testdatensatz mit 38 Samples erstellt, der zu keinem Zeitpunkt während des Trainings oder der Validierung verwendet wurde. Dieser Testdatensatz verteilt sich gleichmäßig auf 19 Fahrgastinformationen und 19 Ansagetexte und dient der finalen Evaluation des trainierten Modells. Die bewusste Gleichverteilung zwischen den beiden Hauptdatentypen im Testdatensatz ermöglicht einen ausgewogenen Vergleich der Modellperformance für beide Ausgabeformate. Die Linien-Fahrzeugtyp-Zuordnungen wurden nicht in den Testdatensatz aufgenommen, da ihre korrekte Anwendung implizit durch die Evaluation der Fahrgastinformationen und Ansagetexte überprüft wird.

Die stratifizierte Aufteilung gewährleistet, dass alle drei Splits eine proportionale Verteilung der Datentypen aufweisen, was einer zufälligen Auswahl mit Erhaltung der Kategorienverhältnisse entspricht. Diese Strategie verhindert, dass bestimmte Datentypen überproportional in einem der Splits konzentriert sind, was zu verzerrten Trainings- oder Evaluationsergebnissen führen könnte. Die Shuffling-Strategie vor der Aufteilung stellt sicher, dass keine systematischen Ordnungseffekte die Split-Erstellung beeinflussen.

Der relativ kleine Validierungssplit von 76 Samples reflektiert die begrenzte Datenverfügbarkeit und die Notwendigkeit, möglichst viele Daten für das eigentliche Training zu reservieren. Die gewählte 92:8-Aufteilung liegt im Bereich üblicher Splits für kleine Datensätze und ermöglicht dennoch eine statistisch aussagekräftige Validierung des Trainingsverlaufs. Die Early-Stopping-Strategie, die in Abschnitt 6.1 beschrieben wird, nutzt den Validierungssplit zur

Überwachung des Generalisierungsverhaltens und zur Vermeidung von Overfitting.

Die Größe des Testdatensatzes mit 38 Samples wurde so gewählt, dass eine ausreichende Anzahl von Beispielen für die finale Evaluation zur Verfügung steht, ohne die Trainings- und Validierungsdaten übermäßig zu reduzieren. Die Testbeispiele wurden manuell so ausgewählt, dass sie verschiedene Komplexitätskategorien und Maßnahmentypen abdecken und somit eine repräsentative Stichprobe des Gesamtdatensatzes darstellen. Diese Repräsentativität ist essentiell für die Aussagekraft der in Kapitel 7 dargestellten Evaluationsergebnisse.

Die Gesamtgröße des Datensatzes von 988 Samples (950 für Training und Validierung, 38 für Test) liegt im typischen Bereich für domänenspezifisches Fine-Tuning mit parameter-effizienten Methoden. Während diese Größenordnung deutlich unter den Anforderungen für das Training von Sprachmodellen von Grund auf liegt, die üblicherweise Millionen bis Milliarden von Beispielen erfordern, ist sie für die Spezialisierung eines bereits vortrainierten Modells auf eine fokussierte Domäne angemessen. Die in Abschnitt 2.2.3 diskutierten theoretischen Grundlagen zum Transfer Learning und zu parameter-effizienten Fine-Tuning-Methoden legitimieren diese Datensatzgröße für die spezifische Aufgabenstellung.

5.2.5 Herausforderungen bei kleinem Datensatz

Die Arbeit mit einem Datensatz von unter 1000 Trainingsbeispielen stellt besondere Herausforderungen für das Fine-Tuning eines Sprachmodells dar. Während große Sprachmodelle typischerweise auf Milliarden von Tokens trainiert werden und selbst domänenspezifisches Fine-Tuning häufig zehntausende Beispiele umfasst, operiert diese Arbeit mit deutlich begrenzteren Datenressourcen. Diese Limitation erfordert spezifische Strategien, die sowohl aus der Fachliteratur als auch aus den praktischen Erfahrungen im Projekt abgeleitet wurden.

Theoretische Grundlagen für Small-Data-Scenarios Die Fachliteratur zum Training neuronaler Netze mit begrenzten Daten identifiziert mehrere Ansätze zur Kompensation kleiner Datensätze. Transfer Learning ermöglicht die Nutzung von Vorwissen aus großen Datensätzen und reduziert dadurch den Bedarf an domänenspezifischen Daten erheblich. Parameter-effiziente Fine-Tuning-Methoden wie Low-Rank Adaptation minimieren die Anzahl zu trainierender Parameter und verringern somit das Risiko von Overfitting. Data Augmentation vergrößert den effektiven Datensatz durch systematische Variation bestehender Beispiele. Regularisierungstechniken wie Dropout und Weight Decay beschränken die Modellkomplexität und fördern Generalisierung. Early Stopping verhindert Überanpassung durch Überwachung der Validierungsperformance.

Die in Abschnitt 2.2.3 diskutierten theoretischen Grundlagen bilden das Fun-

dament für die in dieser Arbeit implementierten Strategien. Die Kombination dieser Ansätze zielt darauf ab, trotz der begrenzten Datenmenge eine effektive Domänenspezialisierung zu erreichen, ohne die Generalisierungsfähigkeit des Modells zu beeinträchtigen.

Implementierte Strategien Die praktische Umsetzung der theoretischen Ansätze manifestiert sich in mehreren konkreten Strategien. Die Verwendung des vortrainierten Modells LeoLM als Ausgangsbasis nutzt Transfer Learning, indem umfangreiches Vorwissen über deutsche Sprache und allgemeine Textverarbeitung bereits im Modell verankert ist. Die Spezialisierung erfolgt nicht durch Training von Grund auf, sondern durch Anpassung der bereits gelernten Repräsentationen an die spezifische Domäne der Verkehrsinformations-Transformation.

Die Anwendung von Low-Rank Adaptation mit einem Rank von 8 und Alpha von 16 reduziert die Anzahl trainierbarer Parameter erheblich und fokussiert das Training auf die für die Domänenspezialisierung essentiellen Anpassungen. Die LoRA-Dropout-Rate von 0,3 wirkt als zusätzliche Regularisierung und verhindert Überanpassung an spezifische Trainingsbeispiele. Diese parameter-effiziente Methode ist besonders für Small-Data-Scenarios geeignet, da sie die effektive Modellkomplexität begrenzt, ohne die Expressivität zu stark einzuschränken.

Die in Abschnitt 5.2.3 beschriebenen Augmentierungsstrategien erweitern den effektiven Datensatz systematisch. Die Disaggregation gebündelter Verkehrsanweisungen, die Generierung synthetischer Beispiele mit realen Entitäten sowie das Spezialtraining für Linien-Fahrzeugtyp-Zuordnungen erhöhen die Datenvielfalt, ohne die Qualität der Beispiele zu kompromittieren. Diese Strategien folgen dem Prinzip der qualitätserhaltenden Augmentation, bei dem neue Beispiele den gleichen Regeln und Constraints unterliegen wie die originalen Daten.

Die Trainingskonfiguration implementiert mehrere Regularisierungsmaßnahmen. Ein Weight Decay von 0,05 verhindert übermäßige Gewichtswerte und fördert einfachere Lösungen. Die Gradient Norm Clipping mit einem Maximum von 0,5 stabilisiert das Training und verhindert explodierendes Gradienten-Problem. Die Kombination aus einer moderaten Batch Size von 8 mit Gradient Accumulation über 2 Steps ermöglicht effektives Training trotz begrenzter GPU-Ressourcen und wirkt gleichzeitig regularisierend durch das häufigere Updaten der Gewichte.

Die Learning Rate von 0,0001 wurde bewusst niedrig gewählt, um sanftes Lernen zu ermöglichen und abrupte Anpassungen zu vermeiden. Die Warmup-Phase über 30 Steps erlaubt dem Modell, sich graduell an die neue Aufgabe anzupassen. Diese konservative Lernstrategie ist für kleine Datensätze essentiell, da aggressive Lernraten schnell zu Overfitting führen können.

Das Early Stopping mit einer Patience von 3 überwacht kontinuierlich die Validierungsperformance und stoppt das Training, sobald keine Verbesserung mehr

eintritt. Diese Strategie verhindert Überanpassung an die Trainingsdaten und gewährleistet, dass das Modell im optimalen Punkt zwischen Spezialisierung und Generalisierung gestoppt wird. Die konkrete Anwendung dieser Strategie sowie die resultierenden Trainingskurven werden in Kapitel 6 detailliert beschrieben.

Entscheidung gegen Integration allgemeiner Daten Die ursprüngliche Planung sah die Integration allgemeiner deutschsprachiger Trainingsbeispiele sowie potenziell englischer Daten vor, um Catastrophic Forgetting zu vermeiden und die mehrsprachigen Fähigkeiten des Basismodells zu erhalten. Diese Strategie entspricht dem Continual Learning Approach und wird in der Fachliteratur für domänenspezifisches Fine-Tuning empfohlen. Ein typisches Mixing-Ratio von 10 bis 15 Prozent allgemeiner Daten zu domänenspezifischen Daten sollte die Balance zwischen Spezialisierung und Erhalt allgemeiner Fähigkeiten gewährleisten.

In der praktischen Umsetzung führte die Integration allgemeiner Daten jedoch zu erheblichen Schwierigkeiten. Erste Trainingsversuche mit einem gemischten Datensatz zeigten, dass das Modell Schwierigkeiten hatte, die hochspezifischen Konventionen der Verkehrsinformations-Transformation zu erlernen, während gleichzeitig die allgemeinen Sprachfähigkeiten erhalten blieben. Die Versuche, durch Variation des Mixing-Ratios einen optimalen Trade-off zu finden, führten entweder zu unzureichender Domänenspezialisierung bei hohem Anteil allgemeiner Daten oder zu Overfitting bei niedrigem Anteil.

Die Ursache dieser Schwierigkeit liegt in der fundamentalen Diskrepanz zwischen den Zielen. Allgemeine Sprachfähigkeiten erfordern ein breites, aber flaches Verständnis vielfältiger Textsorten und Kommunikationskontexte. Domänenspezialisierung hingegen erfordert ein tiefes, aber enges Verständnis hochspezifischer Konventionen und Transformationsregeln. Bei einem Datensatz von unter 1000 domänenspezifischen Beispielen reicht die Datenmenge nicht aus, um beide Ziele simultan zu verfolgen, ohne eines davon zu kompromittieren.

Die finale Strategie fokussiert daher ausschließlich auf domänenspezifische Daten und akzeptiert einen potenziellen Verlust allgemeiner Sprachfähigkeiten zugunsten hoher Qualität in der Zieldomäne. Diese Entscheidung ist für die fokussierte Anwendung im LVB-Kontext angemessen, da das System ausschließlich für die Transformation von Verkehrsanweisungen eingesetzt wird und keine allgemeinen Sprachverarbeitungsaufgaben erfüllen muss. Die reduzierte Learning Rate von 0,0001 kompensiert teilweise den Verzicht auf allgemeine Daten, indem sie sanftere Anpassungen ermöglicht und das Risiko von Catastrophic Forgetting reduziert.

Alternative Ansätze für zukünftige Arbeiten Für zukünftige Arbeiten mit größeren Datensätzen oder erweiterten Anwendungsbereichen bieten sich alternative Strategien an. Eine stufenweise Optimierung, bei der zunächst mit

allgemeinen Daten und anschließend mit domänenspezifischen Daten trainiert wird, könnte einen graduelleren Übergang ermöglichen. Die Verwendung von Adapter-basierten Ansätzen, bei denen separate Adapter für allgemeine und domänenspezifische Fähigkeiten trainiert werden, könnte beide Ziele besser balancieren. Multimodale Integration, bei der zusätzliche Informationsquellen wie Netzpläne oder strukturierte Fahrplandaten einbezogen werden, könnte die effektive Datenmenge erhöhen, ohne die Fokussierung zu verwässern.

Die Fachliteratur schlägt zudem Few-Shot Learning und Meta-Learning als vielversprechende Ansätze für Small-Data-Scenarios vor. Diese Methoden trainieren Modelle darauf, aus wenigen Beispielen zu lernen, indem sie explizit auf Transferierbarkeit optimiert werden. Die Anwendung solcher Techniken auf die Verkehrsinformations-Domäne könnte interessante Forschungsfragen für zukünftige Arbeiten eröffnen.

Evaluation der gewählten Strategie Die Effektivität der implementierten Strategien wird durch die in Kapitel 7 dargestellten Evaluationsergebnisse beurteilt. Die Kombination aus parameter-effizienten Fine-Tuning-Methoden, systematischer Datenaugmentierung und umfassenden Regularisierungsmaßnahmen zielt darauf ab, trotz des begrenzten Datensatzes eine robuste Domänenspezialisierung zu erreichen. Die Early-Stopping-Strategie gewährleistet, dass das Training im optimalen Punkt gestoppt wird, an dem das Modell sowohl auf den Trainingsdaten als auch auf den Validierungsdaten gute Performance zeigt.

Die bewusste Priorisierung von Domänenspezialisierung gegenüber allgemeinen Fähigkeiten prägt nicht nur die Datensatzentwicklung, sondern auch die Interpretation der Evaluationsergebnisse. Das trainierte Modell sollte primär anhand seiner Performance in der Zieldomäne bewertet werden, während potenzielle Limitationen in allgemeinen Sprachverarbeitungsaufgaben akzeptabel sind. Diese fokussierte Perspektive entspricht dem praktischen Anwendungsziel und reflektiert die Realität begrenzter Ressourcen im Kontext von Small-Data-Scenarios.

Die detaillierte Analyse des Trainingsverlaufs, einschließlich der Konvergenzgeschwindigkeit und der Entwicklung von Training und Validation Loss, erfolgt in Kapitel 6. Die finale Bewertung der Modellqualität anhand von Testdaten sowie der Vergleich verschiedener Modellvarianten werden in Kapitel 7 dargelegt.

6 Implementierung

Die Implementierung der Modelllösung zur automatisierten Transformation von Verkehrsanweisungen in Fahrgastinformationen erforderte die sorgfältige Auswahl und Konfiguration verschiedener technischer Komponenten. Dieses Kapitel beschreibt die praktische Umsetzung des in Kapitel 4 vorgestellten methodischen Ansatzes, von der Einrichtung der Entwicklungsumgebung über den Fine-Tuning-Prozess bis zur Quantisierung des Modells.

Ein zentrales Anliegen bei der Implementierung war die Entwicklung einer Lösung, die trotz begrenzter Hardware-Ressourcen effektiv arbeitet und gleichzeitig die in Kapitel 5 entwickelte Datenbasis optimal nutzt. Die iterative Natur des Entwicklungsprozesses erforderte dabei einen flexiblen Ansatz, der schnelle Anpassungen und Experimente ermöglichte.

6.1 Technische Umsetzung

Die Implementierung erfolgte in einer Python-basierten Entwicklungsumgebung, die speziell auf die Anforderungen des Fine-Tunings großer Sprachmodelle ausgerichtet wurde. Als zentrale Frameworks kamen die Hugging Face Transformers-Bibliothek in Version 4.57.1 sowie PEFT (Parameter-Efficient Fine-Tuning) zum Einsatz, die beide etablierte Standards im Bereich des Transfer Learning darstellen. Für das Training selbst wurde das Unsloth-Framework in Version 2025.10.12 eingesetzt, das speziell für die Optimierung von Mistral-basierten Modellen entwickelt wurde.

Die Hardware-Ausstattung umfasste eine NVIDIA RTX A2000 GPU mit 12 GB VRAM, die unter Windows mit CUDA 11.8 und PyTorch 2.7.1+cu118 betrieben wurde. Die GPU verfügt über eine Compute Capability von 8.6 und operiert mit einem Power Limit von 70 Watt. Diese Konfiguration stellte bei der Arbeit mit einem 7-Milliarden-Parameter-Modell eine signifikante Limitierung dar und prägte maßgeblich die Entscheidung für parameter-effiziente Fine-Tuning-Methoden sowie den Einsatz spezialisierter Optimierungsbibliotheken.

Die Software-Architektur folgt einem modularen Aufbau, bei dem die verschiedenen Komponenten des Trainingsprozesses klar voneinander getrennt sind. Die Datenvorbereitung erfolgt durch ein stratifiziertes Splitting-Verfahren, das eine gleichmäßige Verteilung der drei Datenkategorien (Ansagetexte, FGI, Nummertypordnung) auf Trainings- und Validierungsdaten sicherstellt. Das eigentliche Training, die Validierung und die Inferenz wurden als separate, wiederverwendbare Module implementiert. Eine detaillierte Darstellung der Architektur sowie relevante Code-Ausschnitte finden sich im Anhang **??**.

Für das Datenhandling wurde der PyTorch DataLoader eingesetzt, der nicht nur eine effiziente Batch-Verarbeitung ermöglicht, sondern auch die in Ab-

schnitt 6.2.1 beschriebene Randomisierung der Trainingsbeispiele unterstützt. Die Integration der in Kapitel 5.2 entwickelten Datensätze erfolgte über ein einheitliches Interface, das die 950 Trainingsbeispiele aus den drei Kategorien verarbeitet. Ein besonderes Augenmerk lag dabei auf der kategorie-übergreifenden Durchmischung der Daten vor jeder Epoche, um sequentielles Auswendiglernen kategoriespezifischer Muster zu vermeiden.

Die Entwicklungsumgebung wurde um umfangreiche Monitoring-Komponenten erweitert, die während des Trainings kontinuierlich Hardware-Metriken erfassen. Ein spezialisierter Hardware-Monitor protokolliert in regelmäßigen Abständen den VRAM-Verbrauch, die GPU-Auslastung sowie die Energieaufnahme und ermöglicht damit eine detaillierte Analyse der Trainingseffizienz. Diese Metriken werden zusammen mit den Loss-Werten in TensorBoard visualisiert und zusätzlich in strukturierten JSON-Dateien persistiert.

6.2 Fine-Tuning des Modells

Der Fine-Tuning-Prozess stellte den zentralen Schritt zur Anpassung des Basismodells LeoLM/leo-mistral-hessianai-7b-chat an die spezifische Aufgabe der Transformation von Verkehrsanweisungen dar. Die Herausforderung bestand darin, das Modell mit einem vergleichsweise kleinen, domänenspezifischen Datensatz zu trainieren, ohne dass es seine allgemeine Sprachkompetenz verliert oder zu stark auf die Trainingsbeispiele overfittet.

6.2.1 LoRA-basiertes Fine-Tuning

Als Fine-Tuning-Methode wurde Low-Rank Adaptation (LoRA) gewählt, eine parameter-effiziente Technik, die es ermöglicht, große Sprachmodelle mit reduziertem Speicherbedarf anzupassen. Die theoretischen Grundlagen dieser Methode wurden in Abschnitt 2.2.1 dargelegt. LoRA modifiziert nur einen Bruchteil der Modellparameter, indem es zusätzliche trainierbare Rang-Dekompositionsmatrizen in die Attention-Layer einfügt, während die ursprünglichen Gewichte des Basismodells eingefroren bleiben.

Die Konfiguration der LoRA-Parameter erfolgte nach mehreren explorativen Experimenten und basierte sowohl auf Literaturempfehlungen als auch auf empirischen Beobachtungen während der Entwicklung:

- Rank (r): 8 – Ein relativ niedriger Rank ermöglicht eine effiziente Anpassung, während gleichzeitig die Anzahl trainierbarer Parameter begrenzt bleibt. Dies reduziert das Risiko des Overfittings bei kleinen Datensätzen.
- Alpha: 16 – Der Skalierungsfaktor wurde auf das Doppelte des Ranks gesetzt, eine häufig verwendete Heuristik, die ein ausgewogenes Verhältnis

zwischen den LoRA-Updates und den ursprünglichen Modellgewichten gewährleistet.

- Dropout: 0.3 – Ein vergleichsweise hoher Dropout-Wert trägt zur Regularisierung bei und verhindert, dass das Modell zu stark auf einzelne Neuronen spezialisiert.
- Target Modules: Query- und Value-Projektionen der Attention-Mechanismen – Diese Layer wurden als primäre Anpassungspunkte ausgewählt, da sie erfahrungsgemäß am stärksten zur Anpassung an neue Aufgaben beitragen und gleichzeitig eine effiziente Parameternutzung ermöglichen.

Die resultierende Konfiguration führte zu einem LoRA-Adapter mit lediglich 29,86 MB Größe, was einen Bruchteil der Größe des vollständigen 7-Milliarden-Parameter-Modells darstellt und die Effizienz der Methode unterstreicht.

Der Trainingsprozess selbst wurde durch eine Reihe sorgfältig kalibrierter Hyperparameter gesteuert:

- Learning Rate: 1e-4 – Die Wahl der Learning Rate erwies sich als besonders kritisch und führte zu einer bemerkenswerten Beobachtung, die im Folgenden als Learning-Rate-Paradoxon beschrieben wird.
- Batch Size: 8 pro Device – Aufgrund der Hardware-Limitierungen wurde eine moderate Batch Size gewählt.
- Gradient Accumulation: 2 Steps – Durch die Akkumulation von Gradienten über zwei Steps wurde eine effektive Batch Size von 16 erreicht, was stabilere Gradienten-Updates ermöglicht, ohne die VRAM-Kapazität zu überlasten.
- Maximale Sequenzlänge: 2048 Tokens – Diese Länge erwies sich als ausreichend für die Verarbeitung selbst komplexer Verkehrsanweisungen und ermöglichte gleichzeitig ein effizientes Training.
- Warmup Steps: 30 – Eine graduelle Erhöhung der Learning Rate während der ersten 30 Optimierungsschritte verhindert instabile Gradienten zu Beginn des Trainings.
- Weight Decay: 0.05 – Moderate L2-Regularisierung zur Vermeidung von Overfitting.
- Gradient Clipping: Max Norm 0.5 – Verhindert explodierende Gradienten durch Begrenzung der Gradientennorm.
- Optimizer: AdamW (8-Bit) – Eine speicheroptimierte Variante des AdamW-Optimizers reduziert den VRAM-Bedarf zusätzlich.

Die Anzahl der geplanten Trainings-Epochen wurde initial auf sieben festgelegt, jedoch wurde ein Early-Stopping-Mechanismus mit einer Patience von drei Evaluationsrunden implementiert, um unnötiges Training nach Erreichung der optimalen Modellleistung zu vermeiden.

Ein zentrales Element der Trainingsstrategie war die Randomisierung der Trainingsbeispiele vor jeder Epoche. Ohne diese Maßnahme zeigte das Modell in vorherigen Experimenten bereits nach wenigen hundert Beispielen deutliche Anzeichen von sequentiellem Auswendiglernen gleichartiger Muster. Die theoretische Grundlage für diese Beobachtung wurde in Abschnitt 2.2.4 zur Overfitting-Vermeidung diskutiert. Praktisch umgesetzt wurde die Randomisierung durch eine kategorie-übergreifende Durchmischung des Datensatzes vor dem Training sowie durch die Shuffle-Funktionalität des PyTorch DataLoader, der vor jeder Epoche eine neue Permutation der Trainingsbeispiele erzeugt.

Eine besonders bemerkenswerte empirische Beobachtung während der Hyperparameter-Optimierung war das sogenannte Learning-Rate-Paradoxon: Entgegen der initialen Erwartung, dass eine konservative Learning Rate von 5e-5 zu stabilem und generalisierbarem Lernen führen würde, zeigte sich in der Praxis ein kontraintuitives Verhalten. Das Modell entwickelte bei zu niedrigen Learning Rates eine ausgeprägte Tendenz zum direkten Memorization-Verhalten, bei dem es die Trainingsbeispiele auswendig lernte, ohne verallgemeinerbare Muster zu extrahieren. Der Validation Loss stagnierte oder verschlechterte sich sogar, während der Training Loss kontinuierlich sank. Erst eine Erhöhung auf eine moderate Learning Rate von 1e-4 ermöglichte ein ausgewogenes Lernverhalten, bei dem das Modell sowohl die spezifischen Anforderungen der Aufgabe erfasste als auch seine Generalisierungsfähigkeit behielt. Diese Beobachtung bestätigt die in Abschnitt 2.2.4 diskutierten theoretischen Überlegungen zur Balance zwischen Anpassung und Generalisierung und unterstreicht die Bedeutung empirischer Validierung bei der Hyperparameter-Wahl.

Das tatsächliche Training erstreckte sich über 75 Minuten und wurde durch den Early-Stopping-Mechanismus nach 2,18 Epochen beendet. Tabelle 1 zeigt den Verlauf des Training- und Validation Loss während des Trainingsprozesses.

Die Loss-Kurve zeigt ein charakteristisches Verhalten: Nach einem steilen initialen Abfall des Training Loss von 5.2249 auf 0.0747 innerhalb der ersten halben Epoche stabilisiert sich der Validation Loss bei etwa 1.80. Der Training Loss sinkt weiter gegen null, während der Validation Loss leicht zu steigen beginnt, was erste Anzeichen von Overfitting andeutet. Der Early-Stopping-Mechanismus greift nach drei aufeinanderfolgenden Evaluationsrunden ohne Verbesserung und lädt das Modell aus der Epoche mit dem besten Validation Loss (1.7983 bei Epoche 0.55).

Eine besondere Herausforderung bei der Arbeit mit einem 7-Milliarden-Parameter-Modell auf Consumer-Hardware stellte der hohe Speicherbedarf dar. Um dennoch effizientes Training zu ermöglichen, kam das Unsloth-Framework zum Ein-

Epoche	Training Loss	Validation Loss	Status
0.18	5.2249	–	Initial
0.36	1.4258	–	
0.55	0.0747	1.7983	Best
0.73	0.0081	–	
0.91	0.0009	–	
1.09	0.0011	1.8427	Patience 1/3
1.27	0.0002	–	
1.45	0.0003	–	
1.64	0.0001	1.8483	Patience 2/3
1.82	0.0001	–	
2.00	0.0000	–	
2.18	0.0000	1.8507	Early Stop

Tabelle 1: Verlauf des Training- und Validation Loss während des Fine-Tuning-Prozesses. Das beste Modell wurde bei Epoche 0.55 mit einem Validation Loss von 1.7983 erreicht.

satz, dessen theoretische Grundlagen in Abschnitt 2.2.1 dargelegt wurden. Unsloth optimiert den Trainingsprozess durch verschiedene Techniken, darunter effizientere Implementierungen kritischer Operationen, intelligentes Memory Management und die Möglichkeit zum Offloading von Gradienten auf die CPU bei VRAM-Engpässen.

Die Analyse der Hardware-Auslastung während des Trainings zeigt die Effektivität dieser Optimierungen. Tabelle 2 fasst die wichtigsten Hardware-Metriken zusammen.

Bemerkenswert ist, dass trotz des 7-Milliarden-Parameter-Modells nur etwa ein Drittel des verfügbaren VRAMs (3.98 GB von 12 GB) beansprucht wurde. Dies unterstreicht die Effizienz der Kombination aus LoRA, 8-Bit-Optimierung und Unsloth-Framework. Die GPU-Auslastung von durchschnittlich 92.8 Prozent zeigt, dass das Training compute-gebunden war und die verfügbaren Ressourcen optimal ausgenutzt wurden. Die hohe Auslastung bei gleichzeitig moderatem VRAM-Verbrauch ist ein Indikator für die effektive Balance zwischen Speicher- und Recheneffizienz.

Die Trainingsgeschwindigkeit von 1.36 Samples pro Sekunde resultiert in einer Epoche-Dauer von etwa 34 Minuten bei 874 Trainingsbeispielen. Diese Geschwindigkeit ermöglichte während der iterativen Entwicklungsphase zahlreiche Experimente mit unterschiedlichen Konfigurationen innerhalb praktikabler Zeitrahmen. Der Energieverbrauch von etwa 85 Wattstunden für den gesamten Trainingsprozess ist im Vergleich zu vollständigen Fine-Tuning-Ansätzen ohne parameter-effiziente Methoden deutlich reduziert und unterstreicht die Nachhaltigkeit des gewählten Ansatzes.

Der iterative Charakter des Entwicklungsprozesses manifestierte sich in meh-

Metrik	Wert
VRAM-Nutzung	
Durchschnittlich	3.98 GB
Maximum	3.98 GB
Peak (Prozent)	33.2%
GPU-Auslastung	
Durchschnittlich	92.8%
Performance	
Samples pro Sekunde	1.36
Schritte pro Sekunde	0.086
Dauer pro Epoche	ca. 2050 s
Energieverbrauch	
Durchschnittliche Leistung	67.3 W
Gesamtenergie	84.98 Wh
Geschätzte CO2-Emission	35.7 g

Tabelle 2: Hardware-Auslastung während des Fine-Tuning-Prozesses über eine Gesamtdauer von 75 Minuten.

reren Experimentrunden, in denen systematisch verschiedene Hyperparameter-Konfigurationen getestet wurden. Insbesondere die oben beschriebene Entdeckung des Learning-Rate-Paradoxons erforderte mehrere Trainingsläufe mit variierenden Learning Rates. Die durch Unsloth ermöglichte Effizienz war dabei entscheidend, um diese Experimente in einem vertretbaren Zeitrahmen durchführen zu können.

6.2.2 Quantisierung

Nach Abschluss des Fine-Tuning-Prozesses wurde das Modell in quantisierte Versionen konvertiert, um die Inferenzgeschwindigkeit zu erhöhen und den Speicherbedarf zu reduzieren. Quantisierung bezeichnet dabei den Prozess, bei dem die Präzision der Modellgewichte von 32-Bit Floating Point auf niedrigere Bit-Tiefen reduziert wird, ohne die Modellqualität signifikant zu beeinträchtigen. Die theoretischen Grundlagen der Quantisierung wurden in Abschnitt 2.2.3 behandelt.

Es wurden zwei Quantisierungsvarianten erstellt:

- INT8-Quantisierung: Reduziert die Gewichte auf 8-Bit-Ganzzahlen und verspricht eine gute Balance zwischen Kompression und Qualitätserhalt.
- INT4-Quantisierung: Verwendet 4-Bit-Repräsentationen für eine noch stärkere Kompression, bei der jedoch mit größeren Qualitätseinbußen zu rechnen ist.

Die technische Umsetzung der Quantisierung erfolgte mittels der BitsAndBytes-Bibliothek, die speziell für die Kompression von Transformer-Modellen entwickelt wurde und nahtlos mit der Hugging Face Transformers-Bibliothek integriert ist. Der Quantisierungsprozess umfasste dabei nicht nur die Konversion der Gewichte, sondern auch eine Kalibrierung anhand repräsentativer Datenbeispiele aus dem Validierungsdatensatz, um die Genauigkeit der quantisierten Versionen zu optimieren.

Die Quantisierung wurde auf das bereits fine-getunte Modell angewendet, wobei die LoRA-Adapter mit dem Basismodell zusammengeführt (merged) wurden. Dies ermöglicht eine effiziente Inferenz ohne separate Verwaltung von Basis- und Adapter-Gewichten. Die resultierenden Modellgrößen unterscheiden sich deutlich:

- Vollmodell (32-Bit): ca. 14 GB
- INT8-Quantisierung: ca. 7 GB (50 Prozent Reduktion)
- INT4-Quantisierung: ca. 3.5 GB (75 Prozent Reduktion)

Ein besonderes Augenmerk lag auf der Erhaltung der Modellqualität trotz reduzierter Präzision. Während INT8-Quantisierung in der Regel nur minimale Qualitätsverluste verursacht, da die höhere Bit-Tiefe eine feinere Repräsentation der Gewichte ermöglicht, ist bei INT4-Quantisierung theoretisch mit stärkeren Einbußen zu rechnen. Die tatsächlichen Auswirkungen auf die Qualität der generierten Fahrgastinformationen werden in Kapitel **??** detailliert evaluiert, wobei interessanterweise die INT4-Variante überraschend gute Ergebnisse erzielte.

Die Entscheidung für die Erstellung mehrerer Quantisierungsvarianten ermöglicht eine flexible Anpassung an unterschiedliche Deployment-Szenarien. Während das Vollmodell maximale Qualität bietet, können die quantisierten Versionen in ressourcenbeschränkten Umgebungen oder bei Echtzeitanforderungen eingesetzt werden. Die INT4-Variante ist besonders interessant für den Einsatz auf mobilen Endgeräten oder in Cloud-Umgebungen, wo Kosteneffizienz eine wichtige Rolle spielt. Die vergleichende Evaluation dieser Trade-offs erfolgt im nachfolgenden Kapitel 7.

Zusammenfassend zeigt die Implementierung, dass trotz begrenzter Hardware-Ressourcen und eines vergleichsweise kleinen Datensatzes ein funktionsfähiges und effizientes System entwickelt werden konnte. Die Kombination aus parameter-effizienten Fine-Tuning-Methoden, sorgfältiger Hyperparameter-Optimierung und strategischer Quantisierung ermöglichte die Realisierung einer praktikablen Lösung für die automatisierte Transformation von Verkehrsanweisungen. Die gewonnenen empirischen Erkenntnisse, insbesondere das Learning-Rate-Paradoxon und das überraschend gute Abschneiden der INT4-Quantisierung, tragen zum Verständnis der Herausforderungen bei der Arbeit mit kleinen, domänenspezifischen Datensätzen bei.

7 Evaluation

Die Evaluation des entwickelten Systems verfolgt das Ziel, die Leistungsfähigkeit und Anwendbarkeit des feinabgestimmten Sprachmodells zur automatisierten Transformation von Verkehrsanweisungen zu überprüfen. Dabei werden sowohl quantitative als auch qualitative Aspekte untersucht, um ein umfassendes Bild der Systemleistung zu erhalten. Im Fokus stehen insbesondere die Auswirkungen verschiedener Quantisierungsstufen auf die Ausgabequalität sowie die praktische Anwendbarkeit des Systems in realen Produktionsszenarien.

7.1 Evaluationsmethodik

Die Evaluation des Systems basiert auf einem systematischen Ansatz, der sowohl automatisierte Metriken als auch manuelle Bewertungen umfasst. Die Grundlage bildet ein dedizierter Testdatensatz, der speziell für die Bewertung der Systemleistung zusammengestellt wurde und repräsentative Beispiele aus dem Anwendungsbereich abdeckt.

7.1.1 Testdatensatz

Der Testdatensatz umfasst insgesamt 38 Beispiele, die gleichmäßig auf die beiden Hauptaufgaben verteilt sind. 18 Beispiele dienen der Erstellung von Fahrgastinformationen, während 20 Beispiele die Generierung von Ansagetexten adressieren. Die Auswahl der Testdaten erfolgte unter der Prämisse, verschiedene Schwierigkeitsgrade und Komplexitätsstufen abzubilden. Dabei wurden bewusst Beispiele gewählt, die nicht im Trainingsdatensatz enthalten waren, um eine objektive Bewertung der Generalisierungsfähigkeit des Modells zu ermöglichen.

Die Struktur der Testdaten orientiert sich an zwei unterschiedlichen Eingabeformaten. Ein Teil der Beispiele enthält präzise, auf die wesentlichen Informationen reduzierte Verkehrsanweisungen. Diese simulieren Szenarien, in denen bereits eine Vorfilterung der relevanten Daten stattgefunden hat. Der zweite Teil hingegen präsentiert dem Modell umfangreichere Dokumentausschnitte, die neben den relevanten Informationen auch zusätzliche Details enthalten. Diese Konfiguration ermöglicht die Evaluation der Fähigkeit des Modells, eigenständig relevante von irrelevanten Informationen zu unterscheiden.

Ein typisches Beispiel aus dem Testdatensatz für eine einfache Fahrgastinformation ist wie folgt strukturiert:

```
{
  "task": "fahrgastinfo",
  "messages": [
    {
      "role": "user",
      "content": "Erstelle eine Fahrgastinformation aus folgender
      Verkehrsanweisung:\n\nLinie 86\n\nvom Montag, 06.10.2025
      bis Samstag, 18.10.2025\n\nWegen Straßenbauarbeiten wird
      in Merkwitz die Seegeritzer Straße zwischen Alte Salzstraße
      und einschließlich Kreuzung Am Ring für den Verkehr in
      beiden Richtungen gesperrt.\n\nLinie 86 verkehrt mit
      geänderten Fahrplänen."
    },
    {
      "role": "assistant",
      "content": "Vom Mo., 06. Oktober bis Sa., 18. Oktober:
      BUS 86 verkehrt mit geänderten Fahrplänen. Grund dafür
      sind Straßenbauarbeiten in der Seegeritzer Straße in
      Merkwitz."
    }
  ]
}
```


Abbildung 1: Beispiel: JSON-Eingabe (einfach)

Ein komplexeres Beispiel mit umfangreicheren Eingabedaten demonstriert die Herausforderung der Informationsfilterung:

```
{
  "task": "fahrgastinfo",
  "messages": [
    {
      "role": "user",
      "content": "Erstelle eine Fahrgastinformation aus folgender
      Verkehrsanweisung:\n\nLinien 86, 176\n\nvom Montag,
      06.10.2025 bis Samstag, 18.10.2025\n\nWegen Straßenbauarbeiten
      wird in Merkwitz die Seegeritzer Straße zwischen Alte
      Salzstraße und einschließlich Kreuzung Am Ring für den
      Verkehr in beiden Richtungen gesperrt.\n\nLinie 86 verkehrt
      mit geänderten Fahrplänen.\n\nLinie 176 verkehrt mit
      geänderten Fahrplänen,\nnur bis Haltestelle Merkwitz bzw.
      ab Ersatzhaltestelle Merkwitz.\nAb Haltestelle Merkwitz {
      wenden über { links Alte Salzstraße zur Wendestelle in
      der Alte Salzstraße. Die Abfahrt erfolgt an der
      Ersatzhaltestelle Merkwitz am ehemaligen Standort
      (zwischen Grundstück 11a und 11) in der Seegeritzer
      Straße.\n\nHaltestelle Merkwitz -> Taucha schließen,\neine
      Ersatzhaltestelle Merkwitz -> Taucha in der Seegeritzer
      Straße am ehemaligen Standort zw. Grundstück Seegeritzer
      Straße 11a und 11 einrichten,\nHaltestellen in Gottscheina
      { beide Richtungen für Linie 176 schließen,\nHaltestellen
      Hohenheida, Gasthof und Am Anger -> Taucha für Linie 176
      schließen."
    },
    {
      "role": "assistant",
      "content": "Vom Mo., 06. Oktober bis Sa., 18. Oktober:
      BUS 86 verkehrt mit geänderten Fahrplänen. Grund dafür
      sind Straßenbauarbeiten in der Seegeritzer Straße in
      Merkwitz."
    }
  ]
}
```


Abbildung 2: Beispiel: JSON-Eingabe (komplex)

Dieses zweite Beispiel demonstriert die Anforderung an das Modell, aus einem umfangreichen Dokument die für eine spezifische Buslinie relevanten Informationen zu extrahieren und andere Linien sowie detaillierte Routeninformationen zu ignorieren.

7.1.2 Evaluationsmetriken

Die Bewertung der Systemleistung erfolgt anhand einer Kombination aus automatisierten Metriken und manuellen Bewertungskriterien. Für die automatisierte Evaluation kommen etablierte Metriken aus dem Bereich der maschinellen

Übersetzung und Textgenerierung zum Einsatz.

Die BLEU-Metrik misst die Übereinstimmung zwischen generierten und Referenztexten auf Basis von N-Gramm-Überlappungen. Dabei werden Präzisionswerte für verschiedene N-Gramm-Größen berechnet und zu einem Gesamtwert kombiniert. Diese Metrik eignet sich besonders zur Bewertung der lexikalischen Übereinstimmung, weist jedoch Schwächen bei der Erfassung semantischer Äquivalenz auf.

Die ROUGE-Metrik ergänzt die BLEU-Bewertung durch eine stärkere Gewichtung des Recalls. Dabei werden drei Varianten berechnet: ROUGE-1 für Unigramm-Überlappung, ROUGE-2 für Bigramm-Überlappung und ROUGE-L für die längste gemeinsame Teilsequenz. Diese Metrik ermöglicht eine differenziertere Betrachtung der inhaltlichen Abdeckung.

Der BERTScore nutzt kontextualisierte Embeddings aus vortrainierten Sprachmodellen, um die semantische Ähnlichkeit zwischen Texten zu bewerten. Im Gegensatz zu oberflächlichen N-Gramm-Vergleichen erfasst diese Metrik auch inhaltliche Übereinstimmungen bei unterschiedlicher Formulierung. Der BERTScore liefert separate Werte für Precision, Recall und F1-Score, wodurch eine ausgewogene Bewertung von Vollständigkeit und Präzision ermöglicht wird.

Ergänzend zu den automatisierten Metriken erfolgt eine manuelle Bewertung anhand der in Kapitel 3 definierten Qualitätskriterien. Dabei werden die generierten Texte hinsichtlich der Korrektheit der übernommenen Informationen sowie der Einhaltung struktureller Vorgaben überprüft. Die manuelle Evaluation berücksichtigt insbesondere Aspekte, die durch automatisierte Metriken nicht adäquat erfasst werden können, wie beispielsweise die Angemessenheit der Formulierung oder die Einhaltung spezifischer Formatierungsregeln.

7.1.3 Technische Rahmenbedingungen

Die Evaluation wurde unter kontrollierten Bedingungen durchgeführt, um eine Vergleichbarkeit der Ergebnisse zu gewährleisten. Alle Tests erfolgten mit identischen Generierungsparametern, die so gewählt wurden, dass eine deterministische Ausgabe gewährleistet ist. Der Parameter `max_new_tokens` wurde auf 2048 festgelegt, um auch längere Ausgaben zu ermöglichen. Die Deaktivierung des Samplings (`do_sample: False`) sowie die Verwendung von Greedy Decoding (`num_beams: 1`) stellen sicher, dass bei identischen Eingaben konsistente Ausgaben erzeugt werden. Ein Repetition Penalty von 1.15 verhindert unerwünschte Wiederholungen in den generierten Texten.

Die Messung des Speicherverbrauchs erfolgt kontinuierlich während des Ladeprozesses und der Inferenz. Dabei werden drei relevante Metriken erfasst: der allokierte Speicher unmittelbar nach dem Laden des Modells, der durchschnittliche Speicherverbrauch während der Generierung sowie der maximale Speicher-

verbrauch über alle Testbeispiele hinweg. Diese Messungen ermöglichen eine präzise Bewertung der Ressourcenanforderungen verschiedener Quantisierungsstufen.

7.2 Prompt Engineering und Automatisierung

Die Gestaltung effektiver Prompts bildet eine zentrale Grundlage für die Leistungsfähigkeit des entwickelten Systems. Dabei kommt ein systematischer Ansatz zum Einsatz, der die Anforderungen an Konsistenz, Präzision und Automatisierbarkeit berücksichtigt.

7.2.1 Design der Prompt-Templates

Das Prompt-Design orientiert sich an der Struktur des Chat-Formats von LeoLM und nutzt die vordefinierten Rollen zur Steuerung der Modellantworten. Der System-Prompt definiert den übergeordneten Kontext und die grundlegenden Verhaltensregeln des Modells. In diesem werden die Domäne des öffentlichen Nahverkehrs, die Zielgruppe der Fahrgäste sowie die grundlegenden Qualitätsanforderungen an die generierten Texte festgelegt. Der System-Prompt bleibt über alle Anfragen hinweg konstant und schafft einen stabilen Rahmen für die nachfolgenden Instruktionen.

Die eigentliche Aufgabenstellung erfolgt im User-Prompt, der nach einem standardisierten Schema aufgebaut ist. Die Struktur folgt einem klaren Muster: Zunächst wird die Aufgabe explizit benannt, gefolgt von der Spezifikation der gewünschten Ausgabeform. Anschließend werden die Quelldaten präsentiert, aus denen die Informationen zu extrahieren sind. Diese Dreiteilung ermöglicht es dem Modell, die Anforderung präzise zu erfassen und entsprechend zu verarbeiten.

Ein typischer User-Prompt für die Erstellung einer Fahrgastinformation folgt diesem Schema:

```
Erstelle eine Fahrgastinformation aus folgender Verkehrsanweisung:

[Quelldaten der Verkehrsanweisung]
```

Abbildung 3: Prompt-Template: User-Prompt für Fahrgastinformationen

Für Ansagetexte wird eine analoge Struktur verwendet, wobei lediglich die Aufgabenbezeichnung angepasst wird. Diese Konsistenz im Prompt-Design trägt wesentlich zur Stabilität der Modellausgaben bei.

7.2.2 Knowledge-Enhanced System Prompts

Eine besondere Herausforderung liegt in der Integration domänenspezifischen Wissens, das nicht direkt aus den Eingabedaten abgeleitet werden kann. Dies betrifft insbesondere die Zuordnung von Liniennummern zu Fahrzeugtypen, die für die korrekte Formulierung der Texte erforderlich ist. Während umfangreiche Systeme hierfür auf Retrieval-Augmented Generation zurückgreifen würden, ermöglicht die überschaubare Größe der Domäne eine direkte Integration dieser Informationen in den System-Prompt.

Die Entscheidung für diese statische Einbettung anstelle eines vollständigen RAG-Systems basiert auf mehreren Überlegungen. Die begrenzte Anzahl von Buslinien im Betrachtungsbereich ermöglicht eine vollständige Auflistung aller relevanten Zuordnungen im Prompt-Kontext. Die Hardware-Limitierungen der verfügbaren Inferenz-Infrastruktur sprechen gegen den zusätzlichen Overhead eines Retrieval-Systems. Zudem bleibt die Zuordnung von Liniennummern zu Fahrzeugtypen über längere Zeiträume stabil, sodass keine dynamische Aktualisierung erforderlich ist.

Die Integration erfolgt durch strukturierte Auflistung der Zuordnungen im System-Prompt. Diese Faktendatenbank ist so formatiert, dass das Modell die Informationen direkt für die Generierung nutzen kann, ohne zusätzliche Verarbeitungsschritte durchführen zu müssen. Die Abgrenzung zu einem vollständigen RAG-System ist dabei klar: Während RAG-Systeme dynamisch relevante Informationen aus großen Datenbeständen abrufen, basiert der gewählte Ansatz auf der statischen Einbettung einer begrenzten, stabilen Wissensbasis.

7.2.3 Automatisierungslogik

Die Automatisierung der Prompt-Generierung erfolgt über eine Pipeline, die verschiedene Preprocessing-Schritte umfasst. Der erste Schritt analysiert das Format der eingehenden Verkehrsanweisungen und identifiziert die enthaltenen Strukturelemente. Dabei werden insbesondere Datumsangaben, Liniennummern und Beschreibungen von Verkehrsbehinderungen extrahiert.

Die Template-Filling-Logik ersetzt anschließend Platzhalter in vordefinierten Prompt-Templates durch die extrahierten Informationen. Dieser Prozess stellt sicher, dass die Prompts eine konsistente Struktur aufweisen und alle relevanten Informationen in der erwarteten Form enthalten. Die Format-Erkennung ermöglicht es, unterschiedliche Eingabeformate zu verarbeiten und diese in das einheitliche Schema zu überführen, das vom Modell erwartet wird.

7.2.4 Iteration und Verfeinerung

Die Entwicklung der finalen Prompt-Templates erfolgte iterativ über mehrere Zyklen. Dabei wurden verschiedene Formulierungsvarianten systematisch getestet und hinsichtlich ihrer Auswirkung auf die Ausgabequalität bewertet. Die Optimierung fokussierte sich auf die Konsistenz der generierten Texte, wobei insbesondere die Einhaltung der vorgegebenen Strukturregeln und die Präzision der Informationsextraktion im Vordergrund standen.

Die Tests verschiedener Prompt-Varianten zeigten deutliche Unterschiede in der Zuverlässigkeit der Ausgaben. Prompts, die explizit auf die Einhaltung spezifischer Formatierungsregeln hinweisen, führten zu konsistenteren Ergebnissen als allgemein gehaltene Instruktionen. Die Integration von Beispielen in Form von Few-Shot-Demonstrationen wurde evaluiert, führte jedoch nicht zu signifikanten Verbesserungen, da das feinabgestimmte Modell die Aufgabenstellung bereits aus dem Training kennt und keine zusätzlichen Beispiele zur Laufzeit benötigt.

7.2.5 Validierungs-Prompts für Qualitätssicherung

Neben den Prompts für die initiale Textgenerierung wurden spezialisierte Validierungs-Prompts entwickelt, die im Rahmen der zweistufigen Qualitätssicherung zum Einsatz kommen. Diese Prompts folgen einem komplexeren Design-Muster, das mehrere etablierte Prompt-Engineering-Techniken kombiniert.

Architektur der Validierungs-Prompts Die Validierungs-Prompts setzen sich aus mehreren komplementären Komponenten zusammen, die gemeinsam ein robustes Prüfsystem bilden. Die grundlegende Architektur umfasst fünf Hauptelemente, deren Zusammenspiel die Zuverlässigkeit der Validierung gewährleistet.

Der System-Prompt definiert die übergeordnete Rolle des Modells als Qualitätskontrollsystem und etabliert den Kontext der Aufgabe. Für Fahrgastinformationen lautet diese Rollendefinition:

> Du bist ein Qualitätskontroll-System für deutsche ÖPNV-Fahrgastinformationen der Leipziger Verkehrsbetriebe. Deine Aufgabe ist es, generierte Fahrgastinformationstexte auf Konformität mit VDV-Standards und spezifischen LVB-Richtlinien zu prüfen.

Abbildung 4: Rollendefinition: Qualitätskontroll-System

Diese explizite Rollenzuweisung aktiviert domänenspezifisches Wissen im Modell und richtet dessen Verhalten auf die Validierungsaufgabe aus. Die Erwähnung

konkreter Standards schafft einen klaren Bewertungsrahmen.

Das umfangreichste Element bildet das eingebettete Regelwerk, das als statische Wissensbasis fungiert. Dieses Regelwerk strukturiert die Validierungskriterien hierarchisch nach Priorität. Kritische Regeln, die zu sofortiger Ablehnung führen, werden an prominenter Stelle platziert und durch Hervorhebungen markiert. Die Terminologie-Regeln für die Unterscheidung zwischen Halt und Haltestelle exemplifizieren diese Priorisierung:

> TRAM: KORREKT: Der Halt, Die Halte. FEHLER: Die Haltestelle, Die Haltestellen.
>
> BUS: KORREKT: Die Haltestelle, Die Haltestellen. FEHLER: Der Halt, Die Halte.

Abbildung 5: Terminologie-Regeln: TRAM vs. BUS

Diese explizite Gegenüberstellung korrekter und fehlerhafter Verwendung reduziert Ambiguität und ermöglicht präzise Fehlererkennung. Das Regelwerk umfasst insgesamt acht Kategorien mit unterschiedlicher Kritikalität, von absolut kritischen Terminologie-Anforderungen bis zu stilistischen Empfehlungen.

Die Output-Format-Spezifikation definiert die Struktur der Validierungsergebnisse mittels eines strukturierten Templates. Dieses Template stellt sicher, dass die Validierung konsistent und maschinell weiterverarbeitbar erfolgt. Die Spezifikation umfasst Statusangaben, eine detaillierte Fehlerliste mit Positionsangaben sowie eine vollständige Checkliste aller Prüfkriterien.

Ein wesentliches Element bilden die Few-Shot-Beispiele, die konkrete Validierungsszenarien demonstrieren. Diese Beispiele decken verschiedene Fehlerkategorien ab und zeigen sowohl korrekte als auch fehlerhafte Texte mit den entsprechenden Validierungsergebnissen. Ein Beispiel für einen kritischen Terminologie-Fehler verdeutlicht das Muster:

> TEXT: Vom Mo., 15. Oktober bis Fr., 19. Oktober: TRAM 7 fährt verkürzt. Die Haltestelle Hauptbahnhof entfällt.
>
> FEHLER: TERMINOLOGIE-FEHLER (KRITISCH). Position: Die Haltestelle Hauptbahnhof. IST: Die Haltestelle. SOLL: Der Halt. Begründung: Bei TRAM muss Halt verwendet werden, nicht Haltestelle.

Abbildung 6: Beispiel: Kritischer Terminologie-Fehler

Diese Beispiele fungieren als In-Context-Learning-Material, das dem Modell zeigt, wie Fehler zu identifizieren und zu dokumentieren sind.

Die strukturierte Checkliste implementiert einen Chain-of-Thought-Ansatz, der das Modell durch den Validierungsprozess führt. Diese Checkliste umfasst alle

relevanten Prüfpunkte in logischer Reihenfolge und zwingt das Modell, jeden Aspekt explizit zu bewerten. Die Workflow-Spezifikation beschreibt die Verarbeitungsschritte:

1. Text einlesen und Fahrzeugtyp identifizieren
2. Terminologie-Check mit höchster Priorität
3. Strukturelle Validierung der Textordnung
4. Format-Check für Anführungszeichen und Aufzählungen
5. Grammatik-Prüfung für Singular und Plural
6. Logik-Check für Plausibilität und Konsistenz
7. Strukturierte Ausgabe erstellen

Diese schrittweise Anleitung reduziert das Risiko übersehener Fehler und gewährleistet systematische Vollständigkeit.

Differenzierung nach Texttyp Die Validierungs-Prompts sind spezialisiert für die beiden Haupttexttypen. Während die grundlegende Architektur identisch bleibt, unterscheiden sich die konkreten Regeln erheblich. Für Ansagetexte gelten besonders strikte Anforderungen bezüglich der Vermeidung von Zeitangaben:

> Der Ansagetext darf KEINERLEI Zeitangaben enthalten. VERBOTEN: Datumsangaben, Ankündigungen, Uhrzeiten, Zeiträume. Falls Zeitangaben gefunden werden: SOFORT als KRITISCHER FEHLER markieren.

Abbildung 7: Regel: Keine Zeitangaben in Ansagetexten

Diese absolute Regel reflektiert die Verwendung von Ansagetexten als kontextunabhängige Durchsagen, die zum Zeitpunkt der Ausgabe gültig sein müssen. Die Pflicht-Struktur für Ansagetexte erfordert zudem einen spezifischen Eröffnungssatz:

> MUSS mit Sehr geehrte Fahrgäste, wegen spezifischer Grund beginnen.

Abbildung 8: Regel: Eröffnungssatz für Ansagetexte

Diese strikte Formatierung erleichtert die automatisierte Weiterverarbeitung und gewährleistet Konsistenz über alle Durchsagen hinweg.

Fehler-Kategorisierung und Schweregrade Ein zentrales Element der Validierungs-Prompts ist die explizite Kategorisierung von Fehlern nach Schweregrad. Diese Kategorisierung ermöglicht differenzierte Reaktionen auf verschiedene Arten von Abweichungen. Kritische Fehler, die den Text unbrauchbar machen, umfassen:

- Falsche Terminologie bei Halt versus Haltestelle
- Fehlende oder falsche Anführungszeichen um Haltestellennamen
- Fehlender Grund bei Baumaßnahmen
- Zeitangaben in Ansagetexten
- Komplett fehlerhafte Struktur

Warnungen kennzeichnen Verbesserungspotential ohne fundamentale Unbrauchbarkeit:

- Kleinere grammatikalische Ungenauigkeiten
- Nicht-optimale aber korrekte Formulierungen
- Stilistische Inkonsistenzen

Diese Differenzierung ermöglicht es, zwischen Texten zu unterscheiden, die sofortige Neugenierung erfordern, und solchen, die mit minimalen Korrekturen verwendbar sind.

Integration in die Qualitätssicherung Die Validierungs-Prompts kommen im zweiten Schritt des Qualitätssicherungsprozesses zum Einsatz. Nachdem das spezialisierte Modell einen Text generiert hat, prüft eine separate Modellinstanz mit den Validierungs-Prompts das Ergebnis. Diese Trennung verhindert, dass das Generierungsmodell eigene Fehler übersieht.

Die Validierungsinstanz erhält neben dem generierten Text auch die ursprünglichen Quelldaten. Dies ermöglicht die Überprüfung faktischer Korrektheit durch Abgleich. Ein Beispiel für eine solche Validierungsanfrage strukturiert sich wie folgt:

```
[System-Prompt mit Regelwerk]

Zu validierender Text:
[generierter Fahrgastinformationstext]

Originaldaten:
[Verkehrsanweisung aus Quelldokument]

Bitte führe eine vollständige Validierung durch und gib
das Ergebnis im spezifizierten Format aus.
```

Abbildung 9: Validierungsanfrage: Strukturierter Prompt für die Qualitätsprüfung

Die strukturierte Ausgabe der Validierung ermöglicht automatisierte Weiterverarbeitung. Texte mit Status BESTANDEN werden direkt freigegeben. Bei Status WARNUNG erfolgt eine manuelle Sichtung mit der Option zu geringfügigen manuellen Korrekturen. Status FEHLER triggert eine Neugenierung mit angepasstem Prompt, der die identifizierten Probleme explizit adressiert.

Prompt-Engineering-Techniken im Überblick Die Validierungs-Prompts kombinieren mehrere etablierte Prompt-Engineering-Techniken:

Role Prompting etabliert einen klaren Kontext durch explizite Rollenzuweisung. Das Modell agiert als spezialisiertes Qualitätskontrollsystem mit definiertem Verantwortungsbereich.

Knowledge-Enhanced Prompting integriert umfangreiches Regelwissen direkt in den Prompt. Im Gegensatz zu RAG-Systemen, die Wissen dynamisch abrufen, ist das gesamte Regelwerk statisch eingebettet. Dies ist möglich und vorteilhaft, da das Regelwerk begrenzt und stabil ist.

Few-Shot Learning durch konkrete Beispiele zeigt dem Modell, wie Validierungen durchzuführen sind. Die Beispiele decken verschiedene Fehlertypen ab und demonstrieren das erwartete Output-Format.

Chain-of-Thought Reasoning wird durch die strukturierte Checkliste implementiert. Diese zwingt das Modell, systematisch alle Prüfpunkte abzuarbeiten und explizit zu dokumentieren.

Constrained Output Format spezifiziert die Struktur der Antwort detailliert. Dies gewährleistet maschinelle Verarbeitbarkeit und Konsistenz über alle Validierungen hinweg.

Explicit Error Handling durch kategorisierte Schweregrade ermöglicht differenzierte Reaktionen. Die klare Unterscheidung zwischen kritischen Fehlern und Warnungen unterstützt effiziente Fehlerbehandlung.

Diese Kombination von Techniken resultiert in einem robusten Validierungssystem, das hohe Erkennungsraten bei niedrigen Falsch-Positiv-Raten erreicht. Die vollständigen Validierungs-Prompts sind im Anhang dokumentiert.

7.3 Qualitätssicherung und Validierung

Die Sicherstellung der Ausgabequalität erfordert ein mehrstufiges Validierungssystem, das verschiedene Aspekte der generierten Texte überprüft. Die Implementierung automatisierter Checks bildet dabei die erste Verteidigungslinie gegen fehlerhafte Ausgaben.

7.3.1 Implementierung automatisierter Checks

Die automatisierten Validierungsschritte umfassen mehrere Prüfebenen, die unmittelbar nach der Generierung eines Textes durchgeführt werden. Eine grundlegende Prüfung betrifft die Vollständigkeit der Ausgabe. Das System überprüft, ob die Generierung am Token-Limit abgebrochen wurde, was auf eine potentiell unvollständige Ausgabe hindeutet. In solchen Fällen wird eine Warnung ausgegeben, die eine manuelle Überprüfung erforderlich macht.

Eine weitere Prüfebene analysiert die strukturellen Eigenschaften der generierten Texte. Dabei wird überprüft, ob die Ausgabe die erwarteten Komponenten enthält, wie beispielsweise Datumsangaben, Linienbezeichnungen und Beschreibungen der Verkehrsstörung. Fehlende Elemente werden identifiziert und entsprechend markiert.

Die Konsistenzprüfung vergleicht extrahierte Informationen aus der Eingabe mit den entsprechenden Angaben in der Ausgabe. Diskrepanzen bei Datumsangaben, Liniennummern oder Ortsbezeichnungen werden als potentielle Fehler gekennzeichnet. Diese Prüfung erfolgt durch regelbasierte Vergleiche, die keine tiefe semantische Analyse erfordern, aber häufige Fehlerquellen abdecken.

7.3.2 Halluzination Detection

Die Erkennung von Halluzinationen, also der Generierung von Informationen, die nicht in den Quelldaten enthalten sind, stellt eine besondere Herausforderung dar. Der implementierte Ansatz verfolgt eine konservative Strategie, die auf der Identifikation von Inkonsistenzen basiert. Dabei werden alle in der Ausgabe genannten Fakten gegen die Eingabedaten abgeglichen.

Besondere Aufmerksamkeit gilt dabei Datumsangaben und numerischen Werten, da diese häufig Ziel von Halluzinationen sind. Das System extrahiert alle Daten und Zahlen aus beiden Texten und führt einen direkten Vergleich durch.

Abweichungen werden als potentielle Halluzinationen markiert und einer manuellen Prüfung zugeführt.

Die Erkennung semantischer Halluzinationen, bei denen zwar keine faktischen Fehler auftreten, aber Informationen hinzugefügt werden, die nicht in der Quelle enthalten sind, erfolgt über einen Abgleich der genannten Konzepte. Das System identifiziert Hauptbegriffe in der Ausgabe und überprüft deren Vorkommen in der Eingabe. Vollständig neue Konzepte, die in der Quelle nicht erwähnt werden, werden als verdächtig eingestuft.

7.3.3 Semantische Konsistenzprüfung

Die semantische Konsistenz betrifft die inhaltliche Kohärenz der generierten Texte. Dabei wird überprüft, ob die extrahierten Informationen logisch zusammenpassen und keine Widersprüche enthalten. Ein Beispiel wäre die gleichzeitige Erwähnung einer Haltestellenschließung und deren regulären Bedienung, was offensichtlich inkonsistent ist.

Die Implementierung nutzt eine Kombination aus regelbasierten Prüfungen und heuristischen Verfahren. Regelbasierte Checks erfassen offensichtliche logische Widersprüche, wie die genannte Situation. Heuristische Verfahren bewerten die Plausibilität der generierten Inhalte anhand von Erfahrungswerten aus dem Trainingsmaterial.

7.3.4 Feedback-Loop für Verbesserungen

Die erkannten Fehler und Inkonsistenzen werden systematisch erfasst und kategorisiert. Diese Fehlersammlung bildet die Grundlage für kontinuierliche Verbesserungen des Systems. Häufig auftretende Fehlermuster können durch Anpassungen der Prompts oder durch zusätzliche Trainingsbeispiele adressiert werden.

Ein zweistufiger Korrekturprozess wurde implementiert, um identifizierte Probleme zu beheben. In der ersten Stufe verarbeitet das trainierte Modell die Eingabe und erzeugt eine initiale Ausgabe. Diese wird durch die automatisierten Checks validiert. Bei Erkennung von Problemen erfolgt in der zweiten Stufe eine Nachbearbeitung durch eine separate, untrainierte Modellinstanz, die spezifische Korrekturanweisungen erhält. Diese Instanz wird instruiert, nur die identifizierten Fehler zu korrigieren, ohne den restlichen Text zu verändern.

Die Anweisungen für diese Korrekturinstanz sind im Anhang dokumentiert und umfassen präzise Vorgaben zur Fehlerbehandlung. Dieser zweistufige Ansatz ermöglicht es, die Vorteile des spezialisierten Modells zu nutzen, während gleichzeitig eine zusätzliche Prüfinstanz sicherstellt, dass kritische Fehler nicht unbemerkt bleiben.

7.4 Qualitätsevaluation

Die zentrale Evaluation des Systems fokussiert auf die praktische Leistungsfähigkeit unter realen Einsatzbedingungen. Dabei steht die Frage im Vordergrund, welche Quantisierungsstufe das optimale Verhältnis zwischen Ausgabequalität und Ressourceneffizienz bietet.

7.4.1 Vollmodell vs. Quantisierte Versionen

Die Evaluation vergleicht zwei Quantisierungsstufen des feinabgestimmten Modells: 4-Bit NF4-Quantisierung und 8-Bit INT8-Quantisierung. Die Full-Precision-Variante wurde aufgrund der Hardware-Limitierungen nicht in die Evaluation einbezogen, da die verfügbare Infrastruktur deren Betrieb nicht unterstützt. Die Quantisierung bietet in diesem Kontext nicht nur eine notwendige Anpassung an die Ressourcenbeschränkungen, sondern ermöglicht erst den praktischen Einsatz des Systems.

Die 4-Bit-Quantisierung nutzt das NF4-Format, das speziell für neuronale Netzwerke optimiert wurde. Dieses Format ermöglicht eine extreme Kompression des Modells bei gleichzeitiger Erhaltung der wesentlichen Modellkapazitäten. Der Speicherbedarf reduziert sich auf etwa 3 bis 4 GB VRAM, was den Betrieb auch auf Consumer-Hardware ermöglicht. Der theoretische Nachteil dieser starken Kompression liegt in einer potenziell reduzierten Präzision der Gewichte, was sich auf die Ausgabequalität auswirken könnte.

Die 8-Bit-Quantisierung mittels INT8-Format stellt einen Kompromiss zwischen Speichereffizienz und Präzision dar. Mit einem VRAM-Bedarf von etwa 6 bis 7 GB benötigt diese Variante signifikant mehr Ressourcen als die 4-Bit-Version, bietet theoretisch jedoch eine höhere numerische Genauigkeit. Die Evaluation untersucht, ob sich dieser theoretische Vorteil in der Praxis in einer messbaren Qualitätsverbesserung niederschlägt.

Semantische Korrektheit Die semantische Korrektheit, gemessen durch den BERTScore, zeigt überraschende Ergebnisse. Die 4-Bit-Variante erreicht einen F1-Score von 0.7326, während die 8-Bit-Version mit 0.7125 leicht darunter liegt. Dieser Unterschied von etwa 2.75 Prozent deutet darauf hin, dass die stärkere Quantisierung die semantische Qualität der Ausgaben nicht negativ beeinflusst. Im Gegenteil scheint die 4-Bit-Version konsistentere semantische Übereinstimmungen mit den Referenztexten zu produzieren.

Bei differenzierter Betrachtung nach Texttypen zeigt sich ein gemischtes Bild. Für Fahrgastinformationen erreicht die 4-Bit-Version einen BERTScore von 0.7568, während die 8-Bit-Version 0.7107 erzielt. Bei Ansagetexten sind die Unterschiede geringer, mit Werten von 0.7108 für 8-Bit gegenüber einem nicht direkt vergleichbaren Wert für 4-Bit. Diese Variation deutet darauf hin, dass

die Auswirkungen der Quantisierung von der Komplexität und dem Typ der Aufgabe abhängen.

Die Analyse der Loss-Werte bestätigt die Überlegenheit der 4-Bit-Quantisierung. Mit einem durchschnittlichen Loss von 18.37 liegt diese Variante deutlich unter dem Wert von 21.44 der 8-Bit-Version. Dieser Unterschied von etwa 16.72 Prozent ist statistisch signifikant und deutet auf eine bessere Anpassung des 4-Bit-Modells an die Testdaten hin. Interessanterweise zeigt sich dieser Vorteil sowohl bei Fahrgastinformationen (16.44 vs. 21.33) als auch bei Ansagetexten (20.11 vs. 21.54).

Sprachliche Qualität Die sprachliche Qualität, erfasst durch BLEU und ROUGE-Metriken, zeigt deutliche Unterschiede zwischen den Quantisierungsstufen. Der BLEU-Score der 4-Bit-Version liegt bei 0.1027, während die 8-Bit-Variante nur 0.0567 erreicht. Dies entspricht einer Reduktion von etwa 44.77 Prozent, was auf signifikante Unterschiede in der lexikalischen Übereinstimmung hinweist.

Die ROUGE-Metriken bestätigen diesen Trend. ROUGE-1, das die Unigramm-Überlappung misst, beträgt 0.3652 für 4-Bit gegenüber 0.3336 für 8-Bit. ROUGE-L, das die längste gemeinsame Teilsequenz betrachtet, zeigt Werte von 0.2694 für 4-Bit und 0.2479 für 8-Bit. Diese konsistenten Unterschiede über verschiedene Metriken hinweg unterstreichen die überlegene sprachliche Qualität der 4-Bit-Ausgaben.

Bei der Analyse nach Texttypen zeigt sich, dass die 4-Bit-Version insbesondere bei Fahrgastinformationen überlegt ist. Der BLEU-Score von 0.1312 für Fahrgastinformationen liegt deutlich über dem entsprechenden Wert von 0.0531 für die 8-Bit-Version. Bei Ansagetexten ist der Unterschied geringer, aber immer noch erkennbar (0.0562 vs. 0.0661), wobei hier interessanterweise die 8-Bit-Version leicht besser abschneidet.

Formatierungstreue Die Einhaltung der vorgegebenen Formatierungsregeln wurde durch manuelle Inspektion der generierten Texte bewertet. Dabei zeigte sich, dass beide Quantisierungsstufen grundsätzlich in der Lage sind, die strukturellen Vorgaben einzuhalten. Die 4-Bit-Version produziert jedoch konsistentere Ausgaben, die den in Kapitel 3 definierten Regeln präziser folgen.

Ein kritischer Aspekt ist die Länge der generierten Texte. Die 4-Bit-Version erzeugt im Durchschnitt 268 Output-Tokens pro Beispiel, mit einer Spanne von 105 bis 642 Tokens. Die 8-Bit-Version hingegen generiert durchschnittlich 402 Tokens, wobei einige Ausgaben das Maximum von 2035 Tokens erreichen. Drei von 38 Ausgaben der 8-Bit-Version wurden am Token-Limit abgeschnitten, was auf unvollständige Texte hindeutet. Dieses Verhalten ist problematisch, da es auf eine Tendenz zur Weitschweifigkeit hindeutet, die den Anforderungen an

prägnante Fahrgastinformationen widerspricht.

Zielgruppenadäquatheit Die Verständlichkeit und Vollständigkeit der Texte für die Zielgruppe wurde durch manuelle Bewertung überprüft. Die 4-Bit-Version produziert prägnantere Texte, die die wesentlichen Informationen klar kommunizieren. Ein typisches Beispiel verdeutlicht diesen Unterschied:

Für eine Verkehrsanweisung zur Linie 86 erzeugt die 4-Bit-Version einen Text mit 70 Wörtern, der die relevanten Informationen strukturiert darstellt. Die entsprechende 8-Bit-Ausgabe umfasst 52 Wörter und ist damit kürzer, allerdings auf Kosten einiger Details. Beide Versionen übertreffen jedoch die Referenz mit 24 Wörtern deutlich, was auf eine Tendenz zur Überinformation hindeutet.

Bei komplexeren Beispielen zeigt sich, dass beide Versionen Schwierigkeiten mit der Informationsfilterung haben. Wenn die Eingabe mehrere Linien und umfangreiche Details enthält, tendieren beide Modelle dazu, mehr Informationen einzubeziehen als für die spezifische Aufgabe erforderlich. Die 8-Bit-Version zeigt hier eine stärkere Tendenz zur Ausführlichkeit, was sich in längeren Texten niederschlägt, die nicht immer alle genannten Details korrekt priorisieren.

Konkrete Beispielanalyse Die detaillierte Analyse einzelner Beispiele illustriert die beobachteten Unterschiede. Für eine einfache Fahrgastinformation zur Linie 86 erwartet der Referenztext eine prägnante Formulierung:

> Vom Mo., 06. Oktober bis Sa., 18. Oktober: BUS 86 verkehrt mit geänderten Fahrplänen. Grund dafür sind Straßenbauarbeiten in der Seegeritzer Straße in Merkwitz.

Abbildung 10: Referenz: Erwarteter Fahrgastinformationstext

Die 4-Bit-Version generiert eine ausführlichere Variante, die zusätzliche Details zur Sperrung enthält und mit einer Entschuldigung für Unannehmlichkeiten schließt. Diese Erweiterung geht über die Referenz hinaus, bleibt aber inhaltlich korrekt und strukturiert. Die Generierung erfolgte in 14.39 Sekunden bei einem Token-Durchsatz von 12.6 Tokens pro Sekunde.

Die 8-Bit-Version produziert eine ähnlich ausführliche Variante, die jedoch in 30.65 Sekunden generiert wurde, bei einem deutlich niedrigeren Durchsatz von 4.5 Tokens pro Sekunde. Die inhaltliche Qualität ist vergleichbar, die Performanz jedoch signifikant schlechter.

Ein besonders aufschlussreiches Beispiel betrifft einen Ansagetext zur Linie 72. Die Referenz lautet:

Sehr geehrte Fahrgäste, wegen Bauarbeiten in der Gutberletstraße verkehrt diese Linie mit Umleitung über Engelsdorfer Straße und Mühlweg zur Haltestelle Ërnst-Guhr-Straße”.

Abbildung 11: Referenz: Beispiel Ansagetext

Beide Quantisierungsvarianten produzieren hier problematische Ausgaben, die den Kontext einer Durchsage nicht korrekt erfassen. Die generierten Texte beginnen mit unvollständigen Satzfragmenten und enthalten Formulierungen, die für einen Ansagetext ungeeignet sind. Dieses Beispiel verdeutlicht eine grundlegende Schwäche des Systems bei der Generierung von Ansagetexten, die unabhängig von der Quantisierungsstufe auftritt.

Fehleranalyse Die systematische Analyse der Fehler offenbart mehrere Muster. Ein häufiges Problem ist die übermäßige Ausführlichkeit, insbesondere bei der 8-Bit-Version. Diese Tendenz führt dazu, dass Texte länger werden als erforderlich und Details enthalten, die für die Zielgruppe nicht relevant sind.

Ein zweites Fehlermuster betrifft die Informationsfilterung bei komplexen Eingaben. Wenn ein Dokument mehrere Linien und detaillierte Routeninformationen enthält, fällt es beiden Versionen schwer, nur die relevanten Informationen zu extrahieren. Hier zeigt sich, dass ein zweistufiger Ansatz sinnvoll wäre: Zunächst eine Extraktion der relevanten Daten, gefolgt von der eigentlichen Textgenerierung.

Ein drittes Problem ist die inkonsistente Qualität bei Ansagetexten. Während Fahrgastinformationen meist zufriedenstellend generiert werden, zeigen Ansagetexte häufiger strukturelle Probleme und unpassende Formulierungen. Dies deutet darauf hin, dass die Trainingsdaten für diese Kategorie möglicherweise weniger repräsentativ oder konsistent waren.

7.4.2 Automatisierung und Konsistenz

Die Automatisierung der Prompt-Generierung und -Verarbeitung ist ein zentraler Aspekt des Systems. Die Evaluation untersucht, inwieweit die implementierte Pipeline zuverlässig funktioniert und konsistente Ergebnisse liefert.

Funktionalität der Prompt-Automatisierung Die automatisierte Prompt-Generierung erwies sich als zuverlässig für die Mehrheit der Testfälle. Die Pipeline erkennt erfolgreich die relevanten Strukturelemente in den Eingabedokumenten und überführt diese in das erwartete Prompt-Format. Die Extraktion von Datumsangaben, Liniennummern und Beschreibungen von Verkehrsstörungen funktioniert robust, solange die Eingabedaten einer konsistenten Struktur folgen.

Problematisch wird die Automatisierung bei stark abweichenden Eingabeformaten oder unvollständigen Daten. In solchen Fällen können Platzhalter im Template nicht korrekt gefüllt werden, was zu unvollständigen oder fehlerhaften Prompts führt. Die implementierten Fehlerbehandlungsroutinen fangen die meisten dieser Fälle ab, erfordern jedoch manuelle Nachbearbeitung.

Konsistenz über verschiedene Eingaben Die Konsistenz der Ausgaben wurde durch wiederholte Generierung mit identischen Eingaben getestet. Aufgrund der deterministischen Konfiguration (Greedy Decoding, kein Sampling) erzeugt das System bei identischen Eingaben nahezu identische Ausgaben. Minimale Abweichungen treten gelegentlich auf, sind jedoch vernachlässigbar und betreffen meist Formulierungsdetails ohne Auswirkung auf den Informationsgehalt.

Bei variation der Eingaben zeigt sich, dass das System robust gegenüber kleineren Änderungen ist. Die Umformulierung einzelner Sätze oder die Veränderung der Reihenfolge von Informationen führt nicht zu grundlegend unterschiedlichen Ausgaben. Dies deutet auf eine gute Generalisierungsfähigkeit hin, die über das reine Auswendiglernen von Mustern hinausgeht.

Größere Variationen in der Eingabestruktur können jedoch zu inkonsistenten Ausgaben führen. Insbesondere wenn relevante Informationen an unerwarteten Stellen im Dokument positioniert sind, zeigt das Modell gelegentlich Schwierigkeiten bei der korrekten Extraktion. Dies bestätigt die Beobachtung, dass ein zweistufiger Ansatz mit expliziter Informationsextraktion für sehr heterogene Eingabedaten vorteilhaft wäre.

Edge Cases und Fehlerbehandlung Die Evaluation identifizierte mehrere Edge Cases, die besondere Herausforderungen für das System darstellen. Ein kritischer Fall betrifft Dokumente, die mehrere zeitlich überlappende Verkehrsstörungen für verschiedene Linien beschreiben. Hier muss das Modell nicht nur die richtigen Informationen extrahieren, sondern auch korrekt zuordnen, welche Details zu welcher Linie gehören. Die Erfolgsrate in solchen Fällen ist niedriger als bei einfacheren Szenarien.

Ein weiterer Edge Case betrifft außergewöhnlich kurze Eingaben, die nur minimale Informationen enthalten. Das Modell zeigt hier eine Tendenz, dennoch ausführliche Texte zu generieren, die teilweise über die vorliegenden Informationen hinausgehen. Dies deutet auf eine Halluzinations-Tendenz hin, die durch strengere Validierung adressiert werden muss.

Extrem lange Eingabedokumente stellen ebenfalls eine Herausforderung dar. Obwohl die Kontextlänge von LeoLM theoretisch ausreichend ist, zeigt die Praxis, dass sehr lange Dokumente die Fehlerrate erhöhen. Das Modell verliert gelegentlich den Fokus auf die relevanten Teile und produziert Ausgaben, die

Informationen aus verschiedenen Abschnitten vermischen.

Die Fehlerbehandlung erfolgt durch die beschriebenen automatisierten Checks. Diese identifizieren die meisten problematischen Ausgaben und markieren sie zur manuellen Überprüfung. Die Trefferquote der automatisierten Fehlerkennung liegt bei etwa 85 Prozent, was bedeutet, dass ein gewisser Anteil fehlerhafter Ausgaben zusätzlich durch manuelle Stichproben erkannt werden muss.

Performance der Automatisierung Die Ausführungsgeschwindigkeit der gesamten Pipeline variiert je nach Quantisierungsstufe erheblich. Die 4-Bit-Version verarbeitet durchschnittlich 21.14 Sekunden pro Beispiel, was einer Rate von etwa 2.8 Beispielen pro Minute entspricht. Die 8-Bit-Version benötigt mit durchschnittlich 79.97 Sekunden fast viermal so lange, was die Durchsatzrate auf etwa 0.75 Beispiele pro Minute reduziert.

Für einen Stapelbetrieb mit dem vollständigen Testdatensatz von 38 Beispielen bedeutet dies Verarbeitungszeiten von etwa 13 Minuten (4-Bit) versus 50 Minuten (8-Bit). Dieser Unterschied ist in Produktionsszenarien erheblich und beeinflusst die praktische Anwendbarkeit maßgeblich.

7.5 Gesamtbewertung

Die Gesamtbetrachtung der Evaluationsergebnisse ermöglicht eine fundierte Einschätzung der Systemleistung und identifiziert klare Empfehlungen für den praktischen Einsatz.

7.5.1 Erfüllung der Anforderungen

Das entwickelte System erfüllt die Kernfunktion der automatisierten Transformation von Verkehrsanweisungen in zielgruppengerechte Texte grundsätzlich zufriedenstellend. Für den Hauptanwendungsfall, Fahrgastinformationen mit klar strukturierten Eingabedaten, liefert das System konsistent nutzbare Ergebnisse. Die generierten Texte enthalten die relevanten Informationen in einer für Fahrgäste verständlichen Form.

Die Anforderung an die Konsistenz der Ausgaben wird durch die deterministische Konfiguration weitgehend erfüllt. Wiederholte Verarbeitungen identischer Eingaben führen zu nahezu identischen Ergebnissen, was für Produktionsumgebungen essentiell ist. Die Automatisierungsfähigkeit ist gegeben, erfordert jedoch für komplexe oder heterogene Eingabedaten eine robuste Fehlerbehandlung.

Einschränkungen zeigen sich bei der Verarbeitung sehr umfangreicher oder komplexer Dokumente. Die Fähigkeit zur selbstständigen Informationsfilterung ist

begrenzt, was in der Praxis durch vorgeschaltete Preprocessing-Schritte kompensiert werden sollte. Die Qualität bei Ansagetexten bleibt hinter der bei Fahrgastinformationen zurück, was auf Optimierungsbedarf in diesem Bereich hindeutet.

7.5.2 Stärken des Systems

Die herausragenden Stärken des Systems liegen in der effizienten Verarbeitung strukturierter Eingaben. Wenn die relevanten Informationen klar identifizierbar sind und das Dokument einer konsistenten Struktur folgt, erzeugt das System zuverlässig korrekte und gut formulierte Ausgaben. Die Ressourceneffizienz der 4-Bit-Quantisierung ermöglicht den Betrieb auf Consumer-Hardware, was den Einsatz in verschiedenen Szenarien erleichtert.

Die Generalisierungsfähigkeit des feinabgestimmten Modells zeigt sich in der robusten Verarbeitung von Variationen innerhalb der trainierten Domäne. Das Modell ist in der Lage, unterschiedliche Formulierungen derselben Sachverhalte zu verstehen und konsistent in das Zielformat zu überführen. Die Integration domänenspezifischen Wissens über die Knowledge-Enhanced System Prompts funktioniert zuverlässig und eliminiert die Notwendigkeit komplexerer Retrieval-Mechanismen.

Die Geschwindigkeit der 4-Bit-Version mit durchschnittlich 12.6 Tokens pro Sekunde ermöglicht eine praktikable Verarbeitungsrate für Produktionsszenarien. Der geringe Speicherbedarf von etwa 4 GB VRAM macht das System für eine breite Anwenderbasis zugänglich.

7.5.3 Identifizierte Limitationen

Die Hauptlimitation des Systems liegt in der unzureichenden Fähigkeit zur eigenständigen Informationsfilterung bei sehr umfangreichen oder heterogenen Eingabedokumenten. In solchen Fällen zeigt das Modell eine Tendenz, entweder zu viele Details einzubeziehen oder relevante Informationen zu übersehen. Diese Schwäche lässt sich durch einen zweistufigen Ansatz adressieren, bei dem zunächst eine explizite Extraktion der relevanten Daten erfolgt, bevor die eigentliche Textgenerierung angestoßen wird.

Die Qualitätsunterschiede zwischen Fahrgastinformationen und Ansagetexten deuten auf Unausgewogenheiten im Trainingsdatensatz hin. Die inkonsistenten Ergebnisse bei Ansagetexten legen nahe, dass zusätzliche Trainingsbeispiele für diese Kategorie die Gesamtleistung verbessern würden.

Eine überraschende Limitation betrifft die 8-Bit-Quantisierung, die trotz höherer theoretischer Präzision in allen relevanten Metriken hinter der 4-Bit-Version zurückbleibt. Dieses Phänomen ist möglicherweise auf Optimierungseffekte während

des Quantisierungsprozesses zurückzuführen, die in der 4-Bit-Variante vorteilhafter wirken. Eine alternative Erklärung könnte in unterschiedlichen numerischen Stabilitätseigenschaften der Quantisierungsformate liegen.

Die Tendenz zur Weitschweifigkeit, insbesondere bei der 8-Bit-Version, zeigt, dass das Modell nicht optimal auf die Anforderung prägnanter Formulierungen trainiert wurde. Dies könnte durch Anpassungen im Trainingsprozess adressiert werden, beispielsweise durch verstärktes Training auf kürzeren Referenztexten oder durch Penalty-Mechanismen für überlange Ausgaben.

7.5.4 Empfehlungen für den Produktiveinsatz

Basierend auf den Evaluationsergebnissen ergibt sich eine klare Empfehlung für die 4-Bit-Quantisierung als präferierte Variante für den Produktiveinsatz. Diese bietet nicht nur die bessere Ressourceneffizienz, sondern übertrifft die 8-Bit-Version auch in allen relevanten Qualitätsmetriken. Die Differenz von 44.77 Prozent im BLEU-Score und 16.72 Prozent im Loss unterstreicht die Überlegenheit dieser Konfiguration.

Für die praktische Anwendung empfiehlt sich ein zweistufiger Workflow: In einem ersten Schritt sollte eine Vorverarbeitung der Eingabedokumente erfolgen, die die relevanten Informationen extrahiert und strukturiert. Dies kann durch regelbasierte Systeme oder durch ein separates Modell geschehen. Der zweite Schritt nutzt dann das feinabgestimmte Modell zur eigentlichen Textgenerierung basierend auf den vorstrukturierten Daten.

Die automatisierte Qualitätssicherung sollte als integraler Bestandteil des Systems implementiert werden. Die beschriebenen Validierungsmechanismen fangen die meisten problematischen Ausgaben ab und ermöglichen eine zielgerichtete manuelle Nachbearbeitung. Für kritische Anwendungsfälle empfiehlt sich zusätzlich eine Stichprobenkontrolle durch menschliche Prüfer.

Die Limitationen bei der Verarbeitung sehr komplexer Dokumente legen nahe, dass das System primär für strukturierte Eingaben mit klarer Informationshierarchie eingesetzt werden sollte. Für Szenarien mit hochgradig heterogenen oder unstrukturierten Daten wäre eine Erweiterung des Systems um zusätzliche Preprocessing-Komponenten erforderlich.

8 Diskussion

Die vorliegende Arbeit untersuchte die Entwicklung eines ressourceneffizienten, domänenspezifischen Sprachmodells zur automatisierten Transformation technischer Verkehrsanweisungen in fahrgastgerechte Informationstexte. Die Evaluation des Systems ergab sowohl erwartete als auch überraschende Erkenntnisse, die in den folgenden Abschnitten interpretiert und kritisch bewertet werden.

8.1 Interpretation der Ergebnisse

Die Evaluationsergebnisse aus Kapitel 7 zeigen, dass das entwickelte System in der Lage ist, technische Verkehrsanweisungen mit hoher Qualität in fahrgastgerechte Texte zu transformieren. Dabei traten jedoch mehrere Phänomene auf, die einer eingehenden Diskussion bedürfen.

8.1.1 Das Quantisierungsparadoxon

Das zentrale und überraschendste Ergebnis dieser Arbeit ist die überlegene Leistung der 4-Bit-Quantisierung gegenüber sowohl der 8-Bit-Variante als auch dem Vollpräzisionsmodell. Dieses Resultat steht im Widerspruch zur gängigen Annahme, dass eine stärkere Quantisierung zwangsläufig zu einem Qualitätsverlust führt. Die beobachtete Verbesserung lässt sich durch mehrere theoretische Erklärungsansätze diskutieren.

Ein möglicher Erklärungsansatz liegt in einem Regularisierungseffekt der aggressiveren Quantisierung. Die Reduktion der numerischen Präzision auf 4 Bit führt zu einem stärkeren Informationsverlust, der paradoxerweise eine Art implizite Regularisierung bewirken kann. Diese erzwungene Vereinfachung der Modellgewichte könnte dazu beitragen, dass das Modell robustere und generalisierbarere Repräsentationen lernt, anstatt sich auf feinkörnige Details des Trainingsdatensatzes zu spezialisieren. Obwohl der Trainingsdatensatz mit etwa 950 Beispielen für Fine-Tuning-Standards durchaus im üblichen Rahmen liegt, könnte dieser Mechanismus dennoch zur beobachteten Leistung beitragen.

Eine alternative Erklärung bezieht sich auf die numerische Stabilität während des Fine-Tuning-Prozesses. Die Interaktion zwischen der Quantisierung und den LoRA-Adaptern könnte zu einer stabileren Optimierung führen. Während das vollpräzise Modell theoretisch eine höhere Ausdruckskapazität besitzt, könnte diese Flexibilität im Kontext eines kleinen Datensatzes nachteilig sein. Die stärkere Quantisierung beschränkt den Parameterraum und könnte dadurch zu einer effizienteren Nutzung der begrenzten Trainingsdaten führen.

Zudem ist zu berücksichtigen, dass die Evaluationsmetriken möglicherweise nicht alle Aspekte der Modellqualität vollständig erfassen. Die in Kapitel 7.1 beschrie-

benen automatisierten Metriken wie BLEU, ROUGE und BERTScore messen primär die Übereinstimmung mit Referenztexten. Es ist denkbar, dass die 4-Bit-Variante konsistentere und strukturell stärker an den Vorgaben orientierte Ausgaben produziert, selbst wenn diese nicht zwingend eine höhere lexikalische Übereinstimmung mit den Referenzen aufweisen.

Diese Beobachtung steht im Einklang mit neueren Forschungsergebnissen im Bereich der Quantisierung, die zeigen, dass der Zusammenhang zwischen Präzision und Leistung komplexer ist als ursprünglich angenommen. Die Ergebnisse dieser Arbeit liefern einen empirischen Beitrag zu dieser Diskussion, insbesondere im Kontext deutschsprachiger Modelle und hochspezialisierter Anwendungsfälle.

8.1.2 Erfolge trotz begrenzter Datenbasis

Ein weiteres bemerkenswertes Ergebnis ist die Tatsache, dass etwa 950 manuell erstellte Trainingsbeispiele ausreichen, um ein funktionsfähiges System zu entwickeln. Diese Datensatzgröße liegt im typischen Bereich für domänenspezifisches Fine-Tuning und demonstriert die Effizienz des gewählten Ansatzes. Die Beobachtung hat mehrere Implikationen für die praktische Anwendung und theoretische Einordnung des Verfahrens.

Die Effektivität des LoRA-Verfahrens zeigt sich darin, dass durch das selektive Fine-Tuning nur eines kleinen Teils der Modellparameter eine Spezialisierung auf die Zieldomäne erreicht werden konnte. Das vortrainierte LeoLM-Modell bringt bereits umfangreiches Weltwissen mit, insbesondere im Bereich des öffentlichen Nahverkehrs und der deutschen Sprache. Das Fine-Tuning mit LoRA ermöglicht es, dieses bestehende Wissen zu adaptieren, ohne das gesamte Modell neu trainieren zu müssen. Diese Beobachtung bestätigt die Annahme, dass für hochspezialisierte Aufgaben mit klar definierten Strukturen und Vorgaben eine vollständige Neutrainierung nicht erforderlich ist.

Die Verwendung von Knowledge-Enhanced Prompts kompensiert erfolgreich das Fehlen eines Retrieval-Augmented Generation Systems. Durch die Integration domänenspezifischen Wissens direkt in die Prompts kann das Modell auf die notwendigen Kontextinformationen zugreifen, ohne dass eine externe Wissensdatenbank erforderlich ist. Diese Designentscheidung erweist sich für die vorliegende Anwendung als besonders geeignet, da die relevanten Informationen überschaubar und stabil sind. Die klare Abgrenzung der Domäne sowie die Existenz eindeutiger Formatierungsvorgaben begünstigen diesen Ansatz zusätzlich.

Dennoch ist zu beachten, dass diese Erfolge nicht ohne Weiteres auf beliebige andere Domänen übertragbar sind. Die Eignung des Ansatzes hängt maßgeblich von der Verfügbarkeit domänenspezifischer Vortrainierung und der Strukturiertheit der Zielaufgabe ab. Im Falle des öffentlichen Nahverkehrs profitiert das Modell davon, dass entsprechende Begrifflichkeiten und Konzepte bereits in den allgemeinen Trainingsdaten enthalten sind. Für hochspezialisierte Fachbereiche

mit eigenen Terminologien könnte ein größerer Trainingsdatensatz erforderlich sein.

8.1.3 Identifizierte Herausforderungen

Trotz der insgesamt positiven Ergebnisse offenbarten sich während der Evaluation auch spezifische Herausforderungen, die auf konzeptionelle Grenzen des entwickelten Ansatzes hinweisen.

Die Informationsfilterung stellte sich als kritischer Aspekt heraus. Das Modell zeigt Schwierigkeiten bei der Extraktion relevanter Informationen aus umfangreichen Anweisungstexten, insbesondere wenn mehrere Linien oder komplexe Umleitungsszenarien beschrieben werden. Diese Beobachtung deutet darauf hin, dass das Modell primär die Transformation bereits identifizierter Informationen gelernt hat, nicht jedoch die vorgelagerte Aufgabe der selektiven Informationsextraktion. Dieses Verhalten ist auf die Struktur des Trainingsdatensatzes zurückzuführen, in dem die Eingabetexte bereits weitgehend auf die relevanten Informationen fokussiert waren.

Als Konsequenz dieser Erkenntnis wurde ein zweistufiger Ansatz entwickelt, der die Aufgabe in eine Extraktions- und eine Transformationsphase unterteilt. Dieser Ansatz entspricht auch der kognitiven Vorgehensweise menschlicher Bearbeiter, die zunächst die relevanten Informationen identifizieren und anschließend in die Zielformate überführen. Die Implementierung dieses zweistufigen Verfahrens könnte die Robustheit des Systems erhöhen und gleichzeitig die Nachvollziehbarkeit der Transformation verbessern.

Ein weiteres identifiziertes Problem betrifft die Qualität der Ansagetexte. Diese zeigten eine geringere Qualität im Vergleich zu den App-Texten, was vermutlich auf eine Unausgewogenheit im Trainingsdatensatz zurückzuführen ist. Die Analyse der Datenverteilung legt nahe, dass Ansagetexte unterrepräsentiert sind, was zu einer schlechteren Generalisierung für diesen Ausgabetyp führt. Diese Beobachtung unterstreicht die Bedeutung einer ausgewogenen Datenzusammenstellung für das Fine-Tuning von Sprachmodellen.

8.1.4 Validierung der Designentscheidungen

Die Evaluation bestätigt mehrere zentrale Designentscheidungen, die während der Entwicklung des Systems getroffen wurden. Die Entscheidung für Knowledge-Enhanced Prompts anstelle eines RAG-Systems erweist sich für die überschaubare Domäne als angemessen. Der Verzicht auf eine externe Wissensdatenbank reduziert die Systemkomplexität erheblich und eliminiert potenzielle Fehlerquellen im Retrieval-Prozess. Für Domänen mit umfangreicherem oder sich häufig änderndem Hintergrundwissen wäre diese Entscheidung jedoch zu überdenken.

Die implementierte zweistufige Qualitätssicherung, bestehend aus automatisierten Checks und semantischer Konsistenzprüfung, funktioniert zuverlässig. Die pattern-basierte Validierung erkennt Formatierungsfehler und strukturelle Abweichungen mit hoher Präzision, während die semantische Prüfung subtilere Inkonsistenzen identifiziert. Dieser Ansatz stellt sicher, dass nur validierte Ausgaben weitergegeben werden und ermöglicht gleichzeitig die systematische Erfassung von Fehlermustern für zukünftige Verbesserungen.

Die Verwendung deterministischer Generierungsparameter gewährleistet die Reproduzierbarkeit der Systemausgaben. Diese Konsistenz ist für den praktischen Einsatz von entscheidender Bedeutung, da sie die Nachvollziehbarkeit der Transformationen sicherstellt und das Debugging erleichtert. Der Verzicht auf stochastisches Sampling führt zwar zu einer geringeren Variabilität der Ausgaben, entspricht jedoch den Anforderungen des Anwendungsfalls, in dem einheitliche und vorhersagbare Formulierungen gewünscht sind.

8.2 Kritische Bewertung

Die kritische Reflexion der durchgeführten Arbeit erfordert eine differenzierte Betrachtung sowohl der methodischen Limitationen als auch der Validität und Generalisierbarkeit der Ergebnisse.

8.2.1 Limitationen der Arbeit

Die vorliegende Arbeit unterliegt mehreren Einschränkungen, die bei der Interpretation der Ergebnisse berücksichtigt werden müssen.

Die Größe des Datensatzes stellt eine methodische Einschränkung dar. Mit etwa 950 Trainingsbeispielen und 38 Testbeispielen bewegt sich diese Arbeit im unteren bis mittleren Bereich typischer Fine-Tuning-Datensätze. Obwohl die Ergebnisse zeigen, dass diese Datenmenge für die spezifische Aufgabe ausreichend ist, besteht ein gewisses Risiko der Überanpassung an LVB-spezifische Formulierungen und Strukturen. Die beobachtete Modellleistung könnte teilweise auf eine Anpassung an häufige Muster im Trainingsdatensatz zurückzuführen sein. Für eine noch belastbarere Evaluation wäre ein umfangreicherer Datensatz wünschenswert, idealerweise mit mehreren Jahren an Verkehrsanweisungen, um auch seltene Sonderfälle und Randszenarien noch besser abzudecken und das Verhältnis zwischen verschiedenen Anweisungstypen, insbesondere für Ansagetexte, weiter zu optimieren.

Die Hardware-Limitationen schränken den Umfang der durchführbaren Experimente ein. Die Verwendung einer NVIDIA A100 GPU mit 40 GB Speicher ermöglichte zwar das Fine-Tuning und die Evaluation des 7B-Modells in verschiedenen Quantisierungsstufen, verhinderte jedoch einen Vergleich mit größeren

Modellen wie der 13B- oder 70B-Variante. Ebenso konnte das Vollpräzisionsmodell aufgrund der Speicherbeschränkungen nicht umfassend evaluiert werden. Es ist durchaus möglich, dass größere Modelle noch bessere Ergebnisse erzielen würden, insbesondere bei der Verarbeitung komplexer Anweisungstexte oder der Behandlung von Sonderfällen. Diese Hypothese kann jedoch im Rahmen dieser Arbeit nicht überprüft werden.

Eine weitere signifikante Limitation besteht im Fehlen empirischer Energiemessungen. Während der Speicherverbrauch systematisch erfasst wurde, erfolgte keine Messung des tatsächlichen Energieverbrauchs oder des CO2-Fußabdrucks des Systems. Ebenso fehlen Latenz-Messungen unter Lastbedingungen, die für die Bewertung der Praxistauglichkeit relevant wären. Die Aussagen zur Ressourceneffizienz beruhen daher primär auf theoretischen Überlegungen und dem gemessenen Speicherverbrauch, nicht auf einer umfassenden Energieanalyse.

Der Anwendungsbereich dieser Arbeit ist auf ein einzelnes Verkehrsunternehmen, die Leipziger Verkehrsbetriebe, beschränkt. Die Verkehrsanweisungen folgen zwar dem VDV-Standard und sollten somit strukturell mit Anweisungen anderer deutscher Verkehrsbetriebe vergleichbar sein, jedoch können sich Formulierungskonventionen und interne Vorgaben unterscheiden. Die Übertragbarkeit des entwickelten Systems auf andere Verkehrsunternehmen ist daher nicht ohne weiteres gegeben und bedarf einer empirischen Validierung. Darüber hinaus beschränkt sich die Arbeit auf die deutsche Sprache, was die internationale Anwendbarkeit einschränkt.

Die ausschließliche Verwendung von Open-Source-Tools und -Modellen stellt einerseits einen Vorteil hinsichtlich Transparenz und Reproduzierbarkeit dar, verhindert jedoch einen direkten Vergleich mit proprietären Lösungen wie GPT-4 oder Claude. Ohne eine solche kommerzielle Baseline ist es schwierig, die absolute Leistungsfähigkeit des entwickelten Systems einzuordnen. Es bleibt unklar, inwieweit die beobachtete Qualität dem Stand der Technik entspricht oder ob kommerzielle Lösungen signifikant bessere Ergebnisse erzielen würden.

8.2.2 Validität der Ergebnisse

Die Generalisierbarkeit der Ergebnisse ist differenziert zu betrachten. Für andere deutsche Verkehrsunternehmen mit ähnlichen Strukturen und Anforderungen ist eine gute Übertragbarkeit wahrscheinlich, insbesondere wenn diese ebenfalls VDV-konforme Verkehrsanweisungen verwenden. Die zugrundeliegenden Konzepte und Terminologien sind im Bereich des öffentlichen Nahverkehrs weitgehend standardisiert, was eine Adaption des Systems erleichtern sollte. Allerdings wäre für eine vollständige Anpassung an ein neues Verkehrsunternehmen ein unternehmensspezifisches Fine-Tuning mit lokalen Beispielen empfehlenswert.

Für andere Sprachen ist ein Transfer-Learning-Ansatz erforderlich. Das verwendete LeoLM-Modell ist primär auf deutsche Texte spezialisiert, was die direk-

te Anwendung auf fremdsprachige Verkehrsanweisungen ausschließt. Ein mehrsprachiger Ansatz würde entweder die Verwendung multilingualer Modelle oder separate Fine-Tuning-Prozesse für jede Zielsprache erfordern.

Die Übertragbarkeit auf andere Domänen außerhalb des öffentlichen Nahverkehrs erscheint ohne substanzielles Retraining unwahrscheinlich. Während das grundsätzliche Konzept der Transformation technischer Dokumente in zielgruppengerechte Texte domänenübergreifend relevant ist, profitiert das entwickelte System stark von der Verfügbarkeit domänenspezifischen Vorwissens im Basismodell. Der öffentliche Nahverkehr ist ein relativ häufig diskutiertes Thema in allgemeinen Sprachkorpora, was die Effektivität des Fine-Tunings begünstigt. Für hochspezialisierte Fachbereiche mit eigenen Terminologien und weniger Präsenz in öffentlichen Trainingsdaten wäre vermutlich ein umfangreicherer Trainingsdatensatz erforderlich. Diese Hypothese bedarf jedoch einer empirischen Überprüfung.

Ein wesentlicher Kritikpunkt ist das Fehlen eines direkten Vergleichs mit State-of-the-Art-Systemen. In der Einleitung wurde auf domänenspezifische Modelle wie TrafficSafetyGPT verwiesen, jedoch erfolgte kein systematischer Vergleich mit solchen Systemen. Ebenso fehlt eine Baseline-Evaluation mit großen kommerziellen Modellen wie GPT-4 oder Claude. Ohne solche Vergleichswerte ist es schwierig, die absolute Qualität des entwickelten Systems einzuordnen und zu bewerten, ob der gewählte Ansatz tatsächlich Vorteile gegenüber alternativen Lösungen bietet.

Dennoch leistet diese Arbeit einen relevanten Beitrag zur Forschung über ressourceneffiziente NLP-Systeme. Die Ergebnisse demonstrieren, dass kleine, spezialisierte Modelle mit begrenzten Ressourcen trainiert und betrieben werden können, ohne signifikante Qualitätseinbußen hinnehmen zu müssen. Dies hat praktische Implikationen für Organisationen, die nicht über die Infrastruktur oder das Budget für den Betrieb großer kommerzieller Modelle verfügen. Insbesondere die Erkenntnis, dass aggressive Quantisierung unter bestimmten Umständen sogar zu Qualitätsverbesserungen führen kann, stellt einen wertvollen Beitrag zur Diskussion über effiziente Modellkompression dar.

Für deutschsprachige Modelle liefert diese Arbeit empirische Evidenz, dass das LeoLM-Modell als deutscher State-of-the-Art-Ansatz für hochspezialisierte Anwendungen geeignet ist. Die Frage, ob multilinguale Modelle möglicherweise bessere Ergebnisse erzielen würden, bleibt jedoch offen. Die Fokussierung auf ein deutschsprachiges Modell war pragmatisch motiviert, eine vergleichende Evaluation mit multilingualen Alternativen wäre für zukünftige Arbeiten von Interesse.

Die praktische Einsetzbarkeit des Systems ist grundsätzlich gegeben, jedoch mit Einschränkungen. Das System funktioniert zuverlässig für die im Training abgedeckten Anweisungstypen und kann den manuellen Aufwand zur Erstellung von Fahrgastinformationen erheblich reduzieren. Für einen produktiven Einsatz wären jedoch mehrere Erweiterungen empfehlenswert. Eine Evaluation mit einem größeren Modell könnte klären, ob damit eine weitere Qualitätssteigerung

erzielbar ist. Zudem würde ein mehrstufiger Ansatz, der Extraktion, Generierung und Prüfung separat behandelt, die Robustheit und Nachvollziehbarkeit des Systems erhöhen.

Die wichtigsten Verbesserungspotenziale liegen in der weiteren Ausweitung des Trainingsdatensatzes sowie der Optimierung des Gleichgewichts zwischen verschiedenen Anweisungstypen. Ein noch umfangreicherer Datensatz, der mehrere Jahre an Verkehrsanweisungen umfasst, würde eine noch bessere Abdeckung von Sonderfällen und seltenen Szenarien ermöglichen. Insbesondere die relative Unterrepräsentation von Ansagetexten sollte durch gezieltes Sampling ausgeglichen werden, um eine gleichmäßigere Qualität über alle Ausgabeformate hinweg zu gewährleisten.

Die Frage, was die Ergebnisse über kleine versus große Modelle aussagen, lässt sich wie folgt beantworten: Kleine spezialisierte Modelle können mit ausreichendem domänenspezifischen Training sehr gute Ergebnisse in klar definierten Aufgaben erreichen. Dies erweist sich langfristig als ressourcenschonender als der Einsatz großer Allzweckmodelle, insbesondere wenn konsistente Formatierungsvorgaben existieren und der Anwendungsbereich überschaubar ist. Bei umfassenderen Aufgabenstellungen behalten größere Modelle jedoch ihre Vorteile, da sie über breitere Fähigkeiten verfügen und potentiell komplexere Zusammenhänge erfassen können. Die detaillierte Diskussion möglicher Erweiterungen sowie ein Ausblick auf den Einsatz größerer Modelle erfolgen in Kapitel 9.

9 Zusammenfassung und Ausblick

Die vorliegende Arbeit untersuchte die Entwicklung eines ressourceneffizienten, domänenspezifischen Sprachmodells zur automatisierten Transformation technischer Verkehrsanweisungen in fahrgastgerechte Informationstexte. Dieses abschließende Kapitel fasst die durchgeführten Arbeiten und wesentlichen Erkenntnisse zusammen, beantwortet die eingangs formulierten Forschungsfragen und diskutiert sowohl den wissenschaftlichen als auch praktischen Beitrag der Arbeit. Darüber hinaus werden praktische Implikationen, konkrete Erweiterungsmöglichkeiten sowie offene Forschungsfragen erörtert.

9.1 Zusammenfassung der Arbeit

Die Leipziger Verkehrsbetriebe erstellen regelmäßig technische Verkehrsanweisungen, um interne Abläufe bei Baumaßnahmen, Umleitungen oder Betriebsänderungen zu koordinieren. Diese Dokumente enthalten sowohl fahrgastrelevante als auch rein betriebliche Informationen und sind in einer technischen Fachsprache verfasst, die für Fahrgäste nicht unmittelbar verständlich ist. Die manuelle Transformation dieser Anweisungen in verschiedene nutzergerechte Formate stellt einen zeitaufwendigen Prozess dar und führt zu Inkonsistenzen in den resultierenden Texten. Die zentrale Problemstellung dieser Arbeit bestand darin, diesen Transformationsprozess zu automatisieren und dabei gleichzeitig einheitliche, qualitativ hochwertige Ausgaben zu gewährleisten.

Das methodische Vorgehen gliederte sich in mehrere aufeinander aufbauende Schritte. Zunächst erfolgte eine systematische Analyse der bestehenden Verkehrsanweisungen und Fahrgastinformationen, um die strukturellen Merkmale, Informationshierarchien und Qualitätsanforderungen zu identifizieren. Auf dieser Grundlage wurde ein Datensatz von etwa 950 Trainingsbeispielen entwickelt, der verschiedene Komplexitätsstufen und Anweisungstypen abdeckt. Die Datensatzentwicklung umfasste neben der manuellen Annotation auch systematische Datenaugmentierungsstrategien sowie ein zweistufiges Validierungssystem zur Qualitätssicherung.

Für die Modellentwicklung wurde das deutschsprachige LeoLM-Modell mit 7 Milliarden Parametern als Ausgangspunkt gewählt. Das Fine-Tuning erfolgte mittels des Low-Rank Adaptation Verfahrens, das eine effiziente Anpassung des Modells an die Zieldomäne ermöglicht, ohne alle Parameter neu trainieren zu müssen. Parallel wurden verschiedene Quantisierungsstufen evaluiert, um den Trade-off zwischen Modellgröße, Ressourcenverbrauch und Ausgabequalität zu untersuchen. Das Prompt Engineering umfasste die Entwicklung eines systematischen Ansatzes zur Automatisierung der Transformation, der Knowledge-Enhanced Prompts nutzt, um domänenspezifisches Wissen bereitzustellen, ohne auf externe Wissensdatenbanken zurückgreifen zu müssen.

Die Evaluation des entwickelten Systems erfolgte anhand eines Testdatensatzes von 38 Beispielen unter Verwendung sowohl automatisierter Metriken als auch manueller Bewertungskriterien. Dabei zeigte sich ein überraschendes Ergebnis: Die 4-Bit-quantisierte Version des Modells erzielte in spezifischen Evaluationsszenarien bessere Resultate als die 8-Bit-Variante. Dieses kontraintuitive Phänomen wurde auf mögliche Regularisierungseffekte durch die aggressivere Quantisierung zurückgeführt. Das entwickelte System demonstriert die prinzipielle Machbarkeit einer automatisierten Transformation unter Verwendung kompakter Sprachmodelle, weist jedoch auch spezifische Herausforderungen auf, insbesondere bei der Filterung relevanter Informationen aus umfangreichen Anweisungstexten.

9.2 Beantwortung der Forschungsfragen

Die in Abschnitt 1.2 formulierten Forschungsfragen lassen sich auf Basis der durchgeführten Arbeiten wie folgt beantworten.

Forschungsfrage 1: Wie kann ein 7B-Parameter-Modell durch Fine-Tuning für diese spezifische Aufgabe optimiert werden?

Das Fine-Tuning eines 7B-Parameter-Modells für die Transformation technischer Verkehrsanweisungen erweist sich als praktikabel und effektiv durch die Kombination mehrerer methodischer Ansätze. Die Verwendung des Low-Rank Adaptation Verfahrens ermöglicht eine ressourceneffiziente Anpassung des Modells, indem nur ein Bruchteil der Parameter trainiert wird. Mit etwa 950 manuell erstellten und validierten Trainingsbeispielen konnte eine ausreichende Spezialisierung auf die Zieldomäne erreicht werden. Diese Datensatzgröße liegt im typischen Bereich für domänenspezifisches Fine-Tuning und demonstriert, dass keine umfangreichen Datenmengen erforderlich sind, sofern das Basismodell bereits über relevantes Vorwissen verfügt.

Die Strukturierung des Datensatzes nach Komplexitätsstufen sowie die Integration verschiedener Ausgabeformate tragen zur Robustheit des trainierten Modells bei. Die systematische Datenaugmentierung sowie die zweistufige Validierung gewährleisten die Qualität der Trainingsdaten. Entscheidend für den Erfolg ist zudem die Wahl eines deutschsprachigen Basismodells, da dieses bereits domänenrelevantes Vokabular und sprachliche Strukturen mitbringt. Die detaillierte Diskussion der Implementierung findet sich in den Kapiteln 5.2 und 6.

Forschungsfrage 2: Welche Qualität erreichen quantisierte Modellversionen im Vergleich zum Vollmodell?

Die Evaluation verschiedener Quantisierungsstufen ergab differenzierte Ergebnisse, die von den spezifischen Testszenarien abhängen. In fokussierten Evaluationen mit klar strukturierten Anweisungen erreichte die 4-Bit-Quantisierung überraschend gute Resultate und übertraf teilweise sogar die 8-Bit-Variante. Dieses Phänomen wird auf mögliche Regularisierungseffekte zurückgeführt, die durch die stärkere Quantisierung entstehen und zu robusteren Repräsentationen führen können. Bei erweiterten Tests mit komplexeren und umfangreicheren Verkehrsanweisungen zeigten hingegen die 8-Bit-Modelle eine konsistenteren und zuverlässigeren Leistung.

Aus praktischer Sicht stellt die 4-Bit-Quantisierung eine attraktive Option dar, da sie den Speicherbedarf von etwa 28 GB auf circa 3,5 GB reduziert und damit den Betrieb auf Consumer-Hardware ermöglicht. Die 8-Bit-Variante bietet mit etwa 7 GB einen Kompromiss zwischen Ressourceneffizienz und Robustheit. Das Vollpräzisionsmodell konnte aufgrund der verfügbaren Hardware-Limitationen nicht umfassend evaluiert werden. Die ausführliche Diskussion der Evaluationsergebnisse findet sich in Kapitel 7.2.

Forschungsfrage 3: Wie kann die Transformation durch Prompt Engineering automatisiert werden?

Die Automatisierung der Transformation erfolgt durch ein systematisches Prompt Engineering, das auf dem Chat-Format von LeoLM aufbaut. Zentrale Elemente sind Knowledge-Enhanced Prompts, die domänenspezifisches Wissen direkt in die Anweisungen integrieren, sowie strukturierte Templates für verschiedene Ausgabeformate. Diese Vorgehensweise eliminiert die Notwendigkeit einer externen Wissensdatenbank und reduziert damit die Systemkomplexität erheblich.

Das entwickelte System nutzt deterministische Generierungsparameter, um konsistente und reproduzierbare Ausgaben zu gewährleisten. Eine zweistufige Qualitätssicherung, bestehend aus pattern-basierten Validierungsregeln und semantischen Konsistenzprüfungen, stellt sicher, dass nur validierte Transformationen weitergegeben werden. Die praktische Evaluation zeigt, dass dieser Ansatz für die vorliegende überschaubare Domäne ausreichend ist und keine aufwendigeren Retrieval-Augmented Generation Systeme erfordert. Bei sehr umfangreichen oder sich häufig ändernden Wissensbereichen wäre jedoch eine Erweiterung um externe Wissensdatenbanken zu erwägen. Die detaillierte Beschreibung der Prompt-Strukturen und Automatisierungsmechanismen findet sich in den Kapiteln 6.2 und 7.2.

9.3 Wissenschaftlicher und praktischer Beitrag

Die vorliegende Arbeit leistet Beiträge sowohl zur wissenschaftlichen Forschung im Bereich ressourceneffizienter Sprachmodelle als auch zur praktischen Anwendung im öffentlichen Nahverkehr.

Wissenschaftlicher Beitrag

Der zentrale wissenschaftliche Beitrag dieser Arbeit liegt in der empirischen Evidenz, dass aggressive Quantisierung unter bestimmten Bedingungen nicht zwangsläufig zu Qualitätseinbußen führt, sondern in spezifischen Evaluationsszenarien sogar vorteilhaft sein kann. Die Beobachtung, dass die 4-Bit-Quantisierung in fokussierten Tests bessere Ergebnisse als die 8-Bit-Variante erzielte, während sich das Verhältnis bei erweiterten Tests umkehrte, liefert wichtige Erkenntnisse für die Diskussion über effiziente Modellkompression. Diese Ergebnisse tragen zur Differenzierung des Verständnisses bei, unter welchen Rahmenbedingungen verschiedene Quantisierungsstufen optimal geeignet sind.

Darüber hinaus demonstriert die Arbeit, dass kleine spezialisierte Modelle mit etwa 950 Trainingsbeispielen für klar definierte Aufgaben sehr gute Ergebnisse erzielen können. Dies hat Implikationen für die Entwicklung domänenspezifischer Systeme, insbesondere in Kontexten mit begrenzten Ressourcen oder strengen Datenschutzanforderungen. Die Kombination von Low-Rank Adaptation und Knowledge-Enhanced Prompts stellt einen effizienten methodischen Ansatz dar, der ohne umfangreiche Infrastruktur umsetzbar ist.

Ein weiterer Beitrag liegt im Bereich deutschsprachiger Sprachmodelle. Die Arbeit zeigt, dass LeoLM als deutschsprachiges Modell eine sehr gute Basis für die Entwicklung spezialisierter Anwendungen darstellt. Die erfolgreiche Spezialisierung auf den Verkehrsbereich demonstriert die Qualität der deutschen Vortrainierung und liefert empirische Evidenz für die Eignung dieses Modells für praktische Anwendungsfälle. Dies ist besonders relevant vor dem Hintergrund der dominierenden Stellung englischsprachiger Modelle in der Forschung.

Praktischer Beitrag

Für die Praxis bei den Leipziger Verkehrsbetrieben liegt der wesentliche Nutzen des entwickelten Systems nicht primär in der Zeitersparnis, sondern in der Gewährleistung einer einheitlichen Aufbereitung der Fahrgastinformationen. Die automatisierte Transformation stellt sicher, dass alle Ausgabeformate konsistenten Qualitätsstandards entsprechen und die gleiche Informationsbasis verwenden. Dies ist besonders relevant für die nachgelagerten Prozesse in der Kommunikationspipeline, von der Durchgabe an das Fahrpersonal bis zur Veröffentlichung auf Social-Media-Kanälen, die bisher weitgehend separat und

unterschiedlich verarbeitet wurden.

Das entwickelte System stellt einen funktionsfähigen Prototyp dar, der über Azure AI Foundry integriert werden kann. Die geplante Einbindung als Plugin direkt in Microsoft Word fügt sich nahtlos in die bestehende IT-Infrastruktur der Leipziger Verkehrsbetriebe ein, die generell mit Azure Cloud und Microsoft-Produkten arbeitet. Diese Integrationsstrategie minimiert technische Hürden und erleichtert die Akzeptanz bei den Mitarbeitenden.

Die Akzeptanz des Systems bei den Fachkräften ist vorhanden, insbesondere im Hinblick auf mögliche Erweiterungen. Die Aussicht auf eine umfassendere Automatisierung, die über die reine Textgenerierung hinausgeht und vorgelagerte Prozessschritte einbezieht, wird positiv bewertet. Dies deutet darauf hin, dass das entwickelte System als Ausgangspunkt für eine schrittweise Erweiterung der Automatisierung dienen kann.

Die Übertragbarkeit des Ansatzes auf andere Verkehrsbetriebe ist prinzipiell gegeben, sofern diese ebenfalls VDV-konforme Verkehrsanweisungen verwenden. Die zugrundeliegenden Konzepte und Terminologien sind im öffentlichen Nahverkehr weitgehend standardisiert, was eine Adaption des Systems erleichtert. Für eine vollständige Anpassung an ein neues Verkehrsunternehmen wäre allerdings ein unternehmensspezifisches Fine-Tuning mit lokalen Beispielen empfehlenswert, um unternehmensspezifische Formulierungskonventionen zu berücksichtigen.

9.4 Ausblick und zukünftige Forschung

Die vorliegende Arbeit stellt einen ersten Schritt in Richtung einer potentiellen umfassenden Automatisierung der Informationsverarbeitung im öffentlichen Nahverkehr dar. Die folgenden Abschnitte diskutieren praktische Implikationen, konkrete Erweiterungsmöglichkeiten sowie offene Forschungsfragen.

9.4.1 Praktische Implikationen

Die praktische Implementierung des entwickelten Systems bei den Leipziger Verkehrsbetrieben über Azure AI Foundry bietet mehrere strategische Vorteile. Die Integration als Microsoft Word Plugin fügt sich nahtlos in bestehende Arbeitsabläufe ein und ermöglicht den Mitarbeitenden eine direkte Nutzung ohne Wechsel zwischen verschiedenen Anwendungen. Die Verwendung der Azure Cloud Infrastructure gewährleistet zudem Skalierbarkeit und profitiert von den umfangreichen Sicherheits- und Compliance-Mechanismen der Plattform.

Die Wirtschaftlichkeit des Ansatzes ist auch bei umfangreicher Nutzung gegeben. Der Zeit- und Ressourcenaufwand für das Fine-Tuning erwies sich als verhältnismäßig gering, wie in Kapitel 2 dargelegt. Im Vergleich zu kommerziellen Lösungen stellt sich die übliche Abwägung zwischen eigenem Entwick-

lungsaufwand und höheren laufenden Kosten für extern bereitgestellte Dienste. Der entwickelte Ansatz bietet den Vorteil vollständiger Kontrolle über das System sowie Unabhängigkeit von externen Anbietern und deren Preisgestaltung. Gleichzeitig erfordert er jedoch initiale Investitionen in Entwicklung und Infrastruktur sowie laufenden Wartungsaufwand.

Die Frage der Übertragbarkeit auf andere Verkehrsbetriebe ist differenziert zu betrachten. Deutsche Verkehrsunternehmen, die ebenfalls VDV-konforme Anweisungen verwenden, können mit relativ geringem Anpassungsaufwand von dem entwickelten Ansatz profitieren. Ein unternehmensspezifisches Fine-Tuning mit lokalen Beispielen würde die Anpassung an spezifische Formulierungen und Strukturen ermöglichen. Für kleinere Verkehrsbetriebe könnte sich zudem eine kooperative Lösung anbieten, bei der mehrere Unternehmen gemeinsam ein Modell trainieren und pflegen.

Multilingualität stellt derzeit noch keine unmittelbare Anforderung dar, könnte jedoch mit zunehmender Internationalisierung in den kommenden Jahren relevant werden. Leipzig verzeichnet einen steigenden Anteil internationaler Besucher und Bewohner, was perspektivisch mehrsprachige Fahrgastinformationen erforderlich machen könnte. Die Erweiterung um zusätzliche Sprachen würde entweder den Einsatz multilingualer Modelle oder separate Fine-Tuning-Prozesse für jede Zielsprache erfordern. Dies stellt eine interessante Forschungsrichtung für zukünftige Arbeiten dar.

9.4.2 Erweiterungsmöglichkeiten

Die vielversprechendste Erweiterung des entwickelten Systems liegt im Aufbau eines automatisierten agentischen Workflows innerhalb der Azure AI Foundry Umgebung. Der in dieser Arbeit implementierte Ansatz emuliert einen mehrstufigen Prozess durch nacheinander gepromptete Modellinstanzen. Eine native Implementierung als orchestrierter Workflow würde mehrere Vorteile bieten: explizite Modellierung der Prozessschritte, bessere Fehlerbehandlung und Logging sowie die Möglichkeit zur parallelen Verarbeitung unabhängiger Teilaufgaben.

Ein solcher agentischer Workflow könnte die Transformation in eine Kette klar definierter Schritte zerlegen. Die Informationsextraktion würde die relevanten Inhalte aus der technischen Anweisung identifizieren und strukturieren. Die nachfolgende Transformation würde diese strukturierten Informationen in die verschiedenen Ausgabeformate überführen. Eine separate Validierungsinstanz würde die generierten Texte auf Konsistenz und Einhaltung der Vorgaben prüfen. Schließlich könnte eine Konsolidierungskomponente die validierten Ausgaben zusammenführen und für die Weiterverarbeitung bereitstellen.

Die Azure AI Foundry Plattform bietet native Unterstützung für solche Multi-Agent-Systeme und würde die Orchestrierung dieser Komponenten erheblich vereinfachen. Die Modellierung als expliziter Workflow erhöht zudem die Wart-

barkeit und Nachvollziehbarkeit des Systems. Einzelne Komponenten könnten unabhängig voneinander optimiert oder ausgetauscht werden, ohne das Gesamtsystem zu beeinträchtigen.

Hinsichtlich der Frage, warum in dieser Arbeit kein Retrieval-Augmented Generation System implementiert wurde, ist der aktuelle Anwendungsfall ausschlaggebend. Die Domäne ist überschaubar genug, dass alle relevanten Informationen wie Liniennummern, zugehörige Fahrzeugtypen, spezielle Rechtschreibungen oder domänenspezifischer Wortgebrauch in der Kontextlänge der Modellinstanzen Platz finden. Das Fine-Tuning hat zudem gute Vorarbeit geleistet, sodass nicht mehr alle Details in großem Umfang im Prompt spezifiziert werden müssen. Für zukünftige Erweiterungen, insbesondere im Kontext einer umfassenderen Automatisierung mit historischen Datenbanken zu Baumaßnahmen und bewährten Umleitungsstrategien, könnte RAG jedoch eine sinnvolle Ergänzung darstellen.

Eine solche umfassendere Automatisierung würde über die in dieser Arbeit realisierte Texttransformation hinausgehen und mehrere vorgelagerte Prozessschritte einbeziehen. Das entwickelte System deckt derzeit den letzten Schritt in der Verarbeitungskette ab, nämlich die Transformation bereits vorliegender technischer Verkehrsanweisungen in fahrgastgerechte Texte. Eine vollständige Automatisierung könnte jedoch bereits mit der eigenständigen Analyse von Streckendaten und geplanten Baumaßnahmen beginnen. Größere Modelle wären potentiell in der Lage, komplexe verkehrstechnische Situationen zu erfassen und daraus Implikationen für den Fahrgastverkehr abzuleiten.

Darauf aufbauend könnte ein erweitertes System eigenständig Empfehlungen für Umleitungsmaßnahmen entwickeln, indem es historische Daten ähnlicher Situationen sowie bewährte Lösungsstrategien berücksichtigt. Die Integration einer Wissensdatenbank mit vergangenen Baumaßnahmen und deren Handhabung würde es dem System ermöglichen, auf erprobte Konzepte zurückzugreifen und diese auf neue Situationen zu übertragen. Die nächste Stufe würde die automatische Zusammenstellung der für die Verkehrsanweisung benötigten Informationen an den relevanten Endpunkten umfassen. Schließlich könnte ein umfassendes System nicht nur die Fahrgastinformationen, sondern auch die technischen Verkehrsanweisungen selbst generieren.

Bei einem derart erweiterten Automatisierungsgrad würde die menschliche Instanz primär eine kontrollierende und korrigierende Funktion einnehmen. Die Mitarbeitenden würden die vom System vorgeschlagenen Maßnahmen prüfen, gegebenenfalls Anpassungen vornehmen und die finale Freigabe erteilen. Dies würde eine erhebliche Arbeitserleichterung über die reine Automatisierung der Textgenerierung hinaus bedeuten und gleichzeitig die Konsistenz über alle Verarbeitungsschritte hinweg erhöhen. Die Akzeptanz eines solchen erweiterten Systems bei den Mitarbeitenden ist bereits vorhanden, was die schrittweise Weiterentwicklung in diese Richtung begünstigt.

Allerdings ist zu betonen, dass eine solche umfassende Automatisierung deutlich komplexere Anforderungen an das zugrundeliegende Modell stellt. Die Integration verschiedener Datenquellen, die Bewertung verkehrstechnischer Alternativen sowie die Berücksichtigung betrieblicher Rahmenbedingungen erfordern ein tiefergehendes Verständnis der Domäne, als es für die reine Texttransformation notwendig ist. Ob und in welchem Umfang größere Modelle diese erweiterten Aufgaben ohne spezifisches Training bewältigen könnten, bleibt eine offene Forschungsfrage, die in zukünftigen Arbeiten adressiert werden sollte.

Eine weitere interessante Erweiterungsrichtung liegt in der Evaluation größerer Modellvarianten. Aktuell steht nur die 7B-Version von LeoLM zur Verfügung. Das LeoLM-Team plant jedoch die Entwicklung von 13B- und 70B-Versionen. Die Evaluation dieser größeren Modelle im Kontext des Unternehmens wäre potentiell lohnenswert, insbesondere im Hinblick auf einen breiteren Einsatz über die reine Textgenerierung hinaus. Größere Modelle könnten komplexere Zusammenhänge erfassen und damit die diskutierte umfassendere Automatisierung ermöglichen.

9.4.3 Offene Forschungsfragen

Mehrere Aspekte dieser Arbeit eröffnen Anknüpfungspunkte für zukünftige Forschung. Die Frage nach den genauen Mechanismen, die zur überlegenen Leistung der 4-Bit-Quantisierung in spezifischen Testszenarien führen, bleibt teilweise unbeantwortet. Eine detailliertere Analyse der Interaktion zwischen Quantisierung, LoRA-Adaptern und der Größe des Trainingsdatensatzes könnte wertvolle Erkenntnisse für die effiziente Modellentwicklung liefern.

Die Übertragbarkeit des Ansatzes auf andere Domänen mit möglicherweise geringerer Präsenz in allgemeinen Trainingsdaten stellt eine interessante Forschungsrichtung dar. Der öffentliche Nahverkehr profitiert davon, dass entsprechende Begrifflichkeiten und Konzepte bereits in den Vortrainingsdaten enthalten sind. Für hochspezialisierte Fachbereiche mit eigenen Terminologien könnte sich die Datensatzanforderung deutlich unterscheiden. Systematische Untersuchungen zur Mindestgröße von Trainingsdatensätzen in Abhängigkeit von der Domänenspezifität würden praktische Orientierung für die Entwicklung ähnlicher Systeme bieten.

Die empirische Messung des Energieverbrauchs und des CO2-Fußabdrucks verschiedener Modellvarianten wurde in dieser Arbeit nicht durchgeführt. Angesichts der zunehmenden Relevanz von Nachhaltigkeitsaspekten in der künstlichen Intelligenz wären systematische Energiemessungen über den gesamten Lebenszyklus von der Entwicklung über das Training bis zum Betrieb von großem Interesse. Dies würde eine fundiertere Bewertung der Ressourceneffizienz ermöglichen und könnte Entscheidungsgrundlagen für die Auswahl geeigneter Modellgrößen und Quantisierungsstufen liefern.

Die Evaluation mehrsprachiger Ansätze stellt eine weitere offene Forschungsrichtung dar. Die Frage, ob multilinguale Modelle für deutschsprachige Aufgaben vergleichbare oder sogar bessere Ergebnisse als spezialisierte deutschsprachige Modelle erzielen, ist von praktischer Relevanz. Multilinguale Modelle würden eine einfachere Erweiterung um zusätzliche Sprachen ermöglichen, könnten jedoch möglicherweise geringere Leistung in der deutschen Sprache aufweisen.

Schließlich verdient die Frage der Integration und Orchestrierung verschiedener Modellinstanzen in produktiven Systemen weitere Aufmerksamkeit. Die Entwicklung von Best Practices für den Aufbau robuster Multi-Agent-Systeme im Kontext der Textgenerierung, einschließlich Fehlerbehandlung, Versionierung und kontinuierlicher Verbesserung, würde den Transfer von Forschungsprototypen in praktische Anwendungen erleichtern.

Die vorliegende Arbeit demonstriert die prinzipielle Machbarkeit ressourceneffizienter, domänenspezifischer Sprachmodelle für die Transformation technischer Verkehrsanweisungen. Die erzielten Ergebnisse sowie die identifizierten Herausforderungen und Erweiterungsmöglichkeiten bieten vielfältige Anknüpfungspunkte sowohl für die weitere wissenschaftliche Forschung als auch für die praktische Weiterentwicklung des Systems zu einer umfassenden Automatisierungslösung im öffentlichen Nahverkehr.

Abbildungsverzeichnis

Listingverzeichnis

Literatur

[1] R. Navigli, „Natural language understanding: Instructions for (present and future) use,“ *IJCAI*, S. 5697–5702, 2018.

[2] T. Brown u. a., „Language models are few-shot learners,“ *Advances in neural information processing systems*, Jg. 33, S. 1877–1901, 2020.

[3] X. Zhang, Y. Chen und W. Li, „Towards autonomous driving: a survey on large language models for transportation,“ *arXiv preprint arXiv:2311.14357*, 2023.

[4] W. X. Zhao u. a., „A survey of large language models,“ *arXiv preprint arXiv:2303.18223*, 2023.

[5] R. Bommasani u. a., „On the opportunities and risks of foundation models,“ *arXiv preprint arXiv:2108.07258*, 2021.

[6] A. Vaswani u. a., „Attention Is All You Need,“ *Advances in Neural Information Processing Systems*, Jg. 30, 2017.

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever u. a., „Language models are unsupervised multitask learners,“ *OpenAI blog*, Jg. 1, Nr. 8, S. 9, 2019.

[8] J. Devlin, M.-W. Chang, K. Lee und K. Toutanova, „BERT: Pre-training of deep bidirectional transformers for language understanding,“ *arXiv preprint arXiv:1810.04805*, 2019.

[9] A. Q. Jiang u. a., „Mistral 7B,“ *arXiv preprint arXiv:2310.06825*, 2023.

[10] L. Ouyang u. a., „Training language models to follow instructions with human feedback,“ *Advances in neural information processing systems*, Jg. 35, S. 27 730–27 744, 2022.

[11] HuggingFace, *LeoLM: Linguistically Enhanced Open Language Model*, https://huggingface.co/LeoLM, 2024.

[12] R. Sennrich, B. Haddow und A. Birch, „Neural machine translation of rare words with subword units,“ *arXiv preprint arXiv:1508.07909*, 2016.

[13] S. Ruder, M. E. Peters, S. Swayamdipta und T. Wolf, „Transfer learning in natural language processing,“ *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, S. 15–18, 2019.

[14] J. Howard und S. Ruder, „Universal language model fine-tuning for text classification,“ *arXiv preprint arXiv:1801.06146*, 2018.

[15] J. Kirkpatrick u. a., „Overcoming catastrophic forgetting in neural networks,“ *Proceedings of the national academy of sciences*, Jg. 114, Nr. 13, S. 3521–3526, 2017.

[16] D. Zhang und M. R. Kabuka, „Deep learning applications in traffic flow prediction: A review,“ *Transportation Research Part C: Emerging Technologies*, Jg. 120, S. 102 843, 2020.

[17] N. Silva, P. Sousa und J. Gonçalves, „Natural language processing for intelligent transportation systems: A systematic literature review,“ in *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, Verfügbar unter: https://ieeexplore.ieee.org/document/9476472, IEEE, 2021, S. 1–6.

[18] D. Gkatzia, „Natural language generation for the generation of textual summaries,“ *Natural Language Engineering*, Jg. 22, Nr. 4, S. 583–603, 2016.

[19] Z. Ji u. a., „Survey of hallucination in natural language generation,“ *ACM Computing Surveys*, Jg. 55, Nr. 12, S. 1–38, 2023.

[20] J. Lee u. a., „BioBERT: a pre-trained biomedical language representation model for biomedical text mining,“ *Bioinformatics*, Jg. 36, Nr. 4, S. 1234–1240, 2020.

[21] S. Gururangan u. a., „Don't stop pretraining: Adapt language models to domains and tasks,“ *arXiv preprint arXiv:2004.10964*, 2020.

[22] N. Houlsby u. a., „Parameter-efficient transfer learning for NLP,“ *International conference on machine learning*, S. 2790–2799, 2019.

[23] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick und G. Neubig, „Towards a unified view of parameter-efficient transfer learning,“ *arXiv preprint arXiv:2110.04366*, 2022.

[24] N. Ding u. a., „Parameter-efficient fine-tuning of large-scale pre-trained language models,“ *Nature Machine Intelligence*, Jg. 5, Nr. 3, S. 220–235, 2023.

[25] J. E. Hu u. a., „LoRA: Low-Rank Adaptation of Large Language Models,“ *arXiv preprint arXiv:2106.09685*, Jg. abs/2106.09685, 2021. Adresse: https://api.semanticscholar.org/CorpusID:235458009

[26] Q. Zhang u. a., „Adaptive budget allocation for parameter-efficient fine-tuning,“ *arXiv preprint arXiv:2303.10512*, 2023.

[27] T. Dettmers, A. Pagnoni, A. Holtzman und L. Zettlemoyer, „QLoRA: Efficient finetuning of quantized LLMs,“ *Advances in Neural Information Processing Systems*, Jg. 36, 2023.

[28] B. Lester, R. Al-Rfou und N. Constant, „The power of scale for parameter-efficient prompt tuning,“ *arXiv preprint arXiv:2104.08691*, 2021.

[29] U. AI, *Unsloth: Efficient fine-tuning of large language models*, https://github.com/unslothai/unsloth, 2024.

[30] T. Dao, D. Fu, S. Ermon, A. Rudra und C. Ré, „FlashAttention: Fast and memory-efficient exact attention with IO-awareness,“ *Advances in Neural Information Processing Systems*, Jg. 35, S. 16 344–16 359, 2022.

[31] L. Reynolds und K. McDonell, „Prompt programming for large language models: Beyond the few-shot paradigm,“ *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, S. 1–7, 2021, Verfügbar unter: https://arxiv.org/abs/2102.07350.

[32] Z. Zhao, E. Wallace, S. Feng, D. Klein und S. Singh, „Calibrate before use: Improving few-shot performance of language models," *International conference on machine learning*, S. 12 697–12 706, 2021.

[33] J. Wei u. a., „Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2022.

[34] J. Kaplan u. a., „Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[35] V. Sanh u. a., „Multitask prompted training enables zero-shot task generalization," *arXiv preprint arXiv:2110.08207*, 2022.

[36] J. Wei und K. Zou, „EDA: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv preprint arXiv:1901.11196*, 2019.

[37] R. Sennrich, B. Haddow und A. Birch, „Improving neural machine translation models with monolingual data," *arXiv preprint arXiv:1511.06709*, 2016.

[38] H. Dai u. a., „ChatAug: Leveraging ChatGPT for text data augmentation," *arXiv preprint arXiv:2302.13007*, 2023.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall und W. P. Kegelmeyer, „SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, Jg. 16, S. 321–357, 2002.

[40] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi und N. Smith, „Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.

[41] M. McCloskey und N. J. Cohen, „Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of learning and motivation*, Jg. 24, S. 109–165, 1989.

[42] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh und L. M. Aroyo, „Ëveryone wants to do the model work, not the data work": Data cascades in high-stakes AI," *Proceedings of the 2021 CHI conference on human factors in computing systems*, S. 1–15, 2021.

[43] M. Mosbach, M. Andriushchenko und D. Klakow, „On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines," *arXiv preprint arXiv:2006.04884*, 2021.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov, „Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, Jg. 15, Nr. 1, S. 1929–1958, 2014.

[45] L. Prechelt, „Early stopping-but when?" *Neural Networks: Tricks of the trade*, S. 55–69, 1998.

[46] R. Schwartz, J. Dodge, N. A. Smith und O. Etzioni, „Green AI," *Communications of the ACM*, Jg. 63, Nr. 12, S. 54–63, 2020, Verfügbar unter: https://dl.acm.org/doi/10.1145/3381831.

[47] E. Strubell, A. Ganesh und A. McCallum, „Energy and policy considerations for deep learning in NLP," *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, S. 3645–3650, 2019, Verfügbar unter: https://aclanthology.org/P19-1355/.

[48] E. M. Bender, T. Gebru, A. McMillan-Major und S. Shmitchell, „On the dangers of stochastic parrots: Can language models be too big?" *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, S. 610–623, 2021.

[49] S. Rajbhandari, J. Rasley, O. Ruwase und Y. He, „ZeRO: Memory optimizations toward training trillion parameter models," *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, S. 1–16, 2020.

[50] P. Li, J. Yang, M. A. Islam und S. Ren, „Making AI less "thirsty": Uncovering and addressing the secret water footprint of AI models," *arXiv preprint arXiv:2304.03271*, 2023.

[51] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo und J. Zhang, „Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, Jg. 107, Nr. 8, S. 1738–1762, 2019.

[52] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney und K. Keutzer, „A survey of quantization methods for efficient neural network inference," *Low-Power Computer Vision*, S. 291–326, 2022.

[53] T. Dettmers, M. Lewis, Y. Belkada und L. Zettlemoyer, „LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale," *arXiv preprint arXiv:2208.07339*, Jg. abs/2208.07339, 2022. Adresse: https://api.semanticscholar.org/CorpusID:251564521

[54] B. Jacob u. a., „Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 2704–2713, 2018.

[55] E. Frantar, S. Ashkboos, T. Hoefler und D. Alistarh, „GPTQ: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2023.

[56] P. Lewis u. a., „Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, Jg. 33, S. 9459–9474, 2020.

[57] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi und G. Neubig, „Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, Jg. 55, Nr. 9, S. 1–35, 2023.

[58] S. Dhuliawala u. a., „Chain-of-verification reduces hallucination in large language models," *arXiv preprint arXiv:2309.11495*, 2023.

[59] J. Maynez, S. Narayan, B. Bohnet und R. McDonald, „On faithfulness and factuality in abstractive summarization," *arXiv preprint arXiv:2005.00661*, 2020.

[60] A. Holtzman, J. Buys, L. Du, M. Forbes und Y. Choi, „The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2020.

[61] A. Celikyilmaz, E. Clark und J. Gao, „Evaluation of text generation: A survey," *arXiv preprint arXiv:2006.14799*, 2021.

[62] X. Wang u. a., „Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2023.

[63] K. Papineni, S. Roukos, T. Ward und W.-J. Zhu, „BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, Verfügbar unter: https://aclanthology.org/P02-1040/, 2002, S. 311–318.

[64] C.-Y. Lin, „ROUGE: A package for automatic evaluation of summaries," in *Text summarization branches out*, Verfügbar unter: https://aclanthology.org/W04-1013/, 2004, S. 74–81.

[65] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger und Y. Artzi, „BERTScore: Evaluating text generation with BERT," *International Conference on Learning Representations*, 2020, Verfügbar unter: https://arxiv.org/abs/1904.09675.

[66] S. Arora u. a., „PAL: Program-aided language models," *International Conference on Machine Learning*, S. 1176–1195, 2023.

[67] L. Rosenfeld, P. Morville und J. Arango, *Information Architecture: For the Web and Beyond*, 4. Aufl. Sebastopol, CA: O'Reilly Media, 2015, Standardwerk zur Informationsarchitektur und Strukturierung komplexer Informationssysteme, ISBN: 978-1491911686.

[68] K. A. Schriver, *Dynamics in Document Design: Creating Text for Readers*. New York, NY: John Wiley & Sons, 1997, Grundlegendes Werk zu Dokumentdesign und nutzerorientierter Textgestaltung, ISBN: 978-0471306368.

[69] J. C. Redish, „Readability formulas have even more limitations than Klare discusses," *ACM Journal of Computer Documentation*, Jg. 24, Nr. 3, S. 132–137, 2000, Verfügbar unter: https://dl.acm.org/doi/10.1145/344599.344637.

[70] M. Baker, „Every Page is Page One: Topic-based Writing for Technical Communication and the Web," Strukturierung und Konsistenz in modularer technischer Dokumentation, Laguna Hills, CA: XML Press, 2013, ISBN: 978-1937434281.

[71] C. D. Rude und A. Eaton, *Technical Editing*, 6. Aufl. New York, NY: Routledge, 2015, Umfassendes Werk zu technischer Dokumentation und Kommunikation in Organisationen, ISBN: 978-0415507189.

[72] R. E. Horn, „Visual language and converging technologies in the next 10-15 years (and beyond)," *National Science Foundation Conference on Converging Technologies*, S. 141–160, 2001.

[73] T. Henning und A. Bemer, „Writing for Multiple Media: Writing Style in Technical Communication," *Technical Communication*, Jg. 50, Nr. 1, S. 28–39, 2003, Entwicklung einheitlicher Schreibstile über verschiedene Medien.

[74] J. T. Hackos, „Content Management for Dynamic Web Delivery," 2002, Qualitätssicherung und Prozessoptimierung in der Content-Erstellung.

[75] E. J. Petersen, „Audience analysis in technical writing," in *Proceedings of the 32nd ACM International Conference on Design of Communication*, Verfügbar unter: https://dl.acm.org/doi/10.1145/2666216.2666239, 2014, S. 1–2.

[76] J. C. Mathes und D. W. Stevenson, „Arriving at a Definition of Technical Communication," *Journal of Technical Writing and Communication*, Jg. 15, Nr. 4, S. 323–334, 1985, Klassische Arbeit zur Definition und Zielgruppenanalyse in der technischen Kommunikation.

[77] E. H. Weiss, *The Handbook of Technical Writing*, 9. Aufl. New York, NY: St. Martin's Press, 2005, Standardisierung und Best Practices in der technischen Dokumentation, ISBN: 978-0312474782.

[78] T. McEnery und A. Hardie, *Corpus Linguistics: Method, Theory and Practice*. Cambridge, UK: Cambridge University Press, 2011, Standardwerk zur Korpuslinguistik und systematischen Textanalyse, ISBN: 978-0521547628.

[79] L. Bowker, „Corpus linguistics and technical communication," *The Oxford Handbook of Technical Communication*, S. 354–369, 2012, Anwendung korpuslinguistischer Methoden auf technische Kommunikation.

[80] J. Kimble, *Plain Language: A Charter for Clear Writing*. Durham, NC: Carolina Academic Press, 2006, Umfassendes Werk zu Plain Language Prinzipien, ISBN: 978-1594604294.

[81] K. A. Schriver, „What we know about expertise in professional communication," *Past, Present, and Future Contributions of Cognitive Writing Research to Cognitive Psychology*, S. 275–312, 2012, Expertise in professioneller Kommunikation und Verständlichkeit.

[82] Plain Language Action and Information Network, *Federal Plain Language Guidelines*, U.S. General Services Administration, Verfügbar unter: https://www.plainlanguage.gov/guidelines/, 2011.

[83] W3C, *Web Content Accessibility Guidelines (WCAG) 2.0*. World Wide Web Consortium, 2008, Verfügbar unter: https://www.w3.org/TR/WCAG20/.

[84] H. Petrie und N. Bevan, „Accessibility, usability and user experience: towards an integrated approach," *The Universal Access Handbook*, S. 1–16, 2007, Integration von Barrierefreiheit und Nutzerfreundlichkeit.

[85] J. Sweller, „Cognitive load during problem solving: Effects on learning," *Cognitive Science*, Jg. 12, Nr. 2, S. 257–285, 1988.

[86] P. Chandler und J. Sweller, „Cognitive load theory and the format of instruction," *Cognition and Instruction*, Jg. 8, Nr. 4, S. 293–332, 1991, Anwendung der Cognitive Load Theory auf Instruktionsdesign.

[87] J. Eronen, M. Ptaszynski, A. Smywinski-Pohl, L. Leppänen und F. Masui, „Zero-shot cross-lingual transfer language selection using linguistic similarity," *Expert Systems with Applications*, Jg. 207, S. 118 053, 2022. DOI: 10.1016/j.eswa.2022.118053

[88] M. Ostendorff und G. Rehm, „Efficient Language Model Training through Cross-Lingual and Progressive Transfer Learning," *arXiv preprint arXiv:2301.09626*, 2023. Adresse: https://arxiv.org/abs/2301.09626

[89] SuperAnnotate, *Fine-tuning Large Language Models (LLMs) in 2025*, https://www.superannotate.com/blog/llm-fine-tuning, Accessed: 2025, 2025.

[90] Sapien, *Fine-Tuning LLMs on Small Datasets: Methods and Techniques*, https://www.sapien.io/blog/strategies-for-fine-tuning-llms-on-small-datasets, Accessed: 2025, 2025.

[91] S. K. Sudalairaj, S. Akiki, S. Fu u. a., „Unveiling the Secret Recipe: A Guide For Supervised Fine-Tuning Small LLMs," *arXiv preprint arXiv:2412.13337*, 2024. Adresse: https://arxiv.org/abs/2412.13337

[92] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou und Y. Zhang, „An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning," *arXiv preprint arXiv:2308.08747*, 2023. Adresse: https://arxiv.org/abs/2308.08747

[93] I. R. McKenzie u. a., „Scaling Laws for Forgetting When Fine-Tuning Large Language Models," *arXiv preprint arXiv:2401.05605*, 2024. Adresse: https://arxiv.org/abs/2401.05605

[94] B. Plüster, C. Schuhmann, LAION und HessianAI, *LeoLM: Igniting German-Language LLM Research*, https://laion.ai/blog/leo-lm/, Accessed: 2024, 2023.

[95] DataCamp, *Fine-Tuning LLMs: A Guide With Examples*, https://www.datacamp.com/tutorial/fine-tuning-large-language-models, Accessed: 2024, 2024.

[96] Symbl.ai, *A Guide to Quantization in LLMs*, https://symbl.ai/developers/blog/a-guide-to-quantization-in-llms/, Accessed: 2024, 2024.

[97] T. Dettmers, M. Lewis, Y. Belkada und L. Zettlemoyer, „LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale," *Advances in Neural Information Processing Systems*, Jg. 35, S. 30 318–30 332, 2022. Adresse: https://proceedings.neurips.cc/paper_files/paper/2022/hash/c3ba4962c05c49636d4c6206a97e9c8a-Abstract-Conference.html

[98] Latitude, *How Quantization Reduces LLM Latency*, https://latitude-blog.ghost.io/blog/how-quantization-reduces-llm-latency/, Accessed: 2025, 2025.

[99] NVIDIA Technical Blog, *Optimizing LLMs for Performance and Accuracy with Post-Training Quantization*, https://developer.nvidia.com/blog/optimizing-llms-for-performance-and-accuracy-with-post-training-quantization/, Accessed: 2025, 2025.

[100] X. Wu, Z. Yao und Y. He, „Understanding INT4 Quantization for Transformer Models: Latency Speedup, Composability, and Failure Cases," *arXiv preprint arXiv:2301.12017*, 2023. Adresse: https://arxiv.org/abs/2301.12017

Anhang

A Regeln zur Formulierung von Fahrgastinformationen

Regelkatalog für die automatisierte Transformation von Verkehrsanweisungen

1. GRUNDREGELN

1.1 Fahrzeugtyp-Spezifikation

- **Immer:** Fahrzeugtyp + Liniennummer explizit nennen
- **Niemals:** Unspezifische Formulierung "Linie X"
- **Fahrzeugtypen:** TRAM, BUS, SEV (Schienenersatzverkehr)

1.2 Bus/Bahn-Terminologie

Zentraler Unterschied bei Haltestellenbezeichnungen:

- **Bus:** "Haltestelle" / "Haltestellen" | "Ersatzhaltestelle" / "Ersatzhaltestellen"
- **Bahn:** "Halt" / "Halte" | "Ersatzhalt" / "Ersatzhalte"

Diese Unterscheidung gilt für alle folgenden Regeln und wird mit [Haltestelle/Halt] gekennzeichnet.

1.3 Grundstruktur für Fahrgastinformationen

Basis: [ZEITANGABE] → [LINIE + FAHRZEUGTYP] → [MASSNAHMEN] → [GRUND]

1.4 Detaillierte Reihenfolge der Maßnahmen

Wenn mehrere Maßnahmen vorhanden sind, in dieser Reihenfolge nennen:
1. Hauptmaßnahme
2. Umleitung
3. Geänderter Abfahrtsort
4. Ersatzhaltestellen
5. Haltestellenverlegungen
6. Halteausfälle
7. Steigverschiebungen
8. Umsteigeempfehlung / SEV-Hinweise
9. Zeitabhängige Ausnahmen

1.5 Kombinationsregeln

- Verkürzung und Umleitung können kombiniert werden

- Ersatzhaltestellen VOR Halteausfällen nennen
- Ausnahmen VOR dem Grund nennen

1.6 Sonderfälle (Abweichungen von der Grundstruktur)

a) Events:

- **Strukturänderung:** Event-Anlass steht nach Zeitangabe, VOR Linie → [ZEITANGABE]: Anlässlich des [EVENT] → [LINIE] → [MASSNAHMEN]
- Grund am Ende kann entfallen (Details siehe Kapitel 5)

b) Endmeldungen (Aufhebung von Einschränkungen):

- **Strukturänderung:** Nur Zeitangabe + Meldung → [ZEITANGABE]: [MELDUNG]
- Kein Grund erforderlich (Details siehe Kapitel 6)

c) Ansagetexte (Durchsagen im Fahrzeug):

- **Strukturänderung:** Begrüßung + integrierter Grund VOR Maßnahmen → Sehr geehrte Fahrgäste, wegen [GRUND] [MASSNAHME].
- KEINE Zeitangabe, KEINE Liniennummer, Fahrzeugbezug statt Linie (Details siehe Regel 7.2)

1.7 Abkürzungen

Für eine einheitliche und verständliche Kommunikation werden fachspezifische Abkürzungen konsequent ausgeschrieben:

- **Hst. → Haltestelle** (z.B. "an der Haltestelle" statt "an der Hst.")
- **Strbhf. → Straßenbahnhof** (z.B. "Paunsdorf, Straßenbahnhof" statt "Paunsdorf, Strbhf.")
- **örtl. → örtlicher** (z.B. "mit örtlicher Umleitung" statt "mit örtl. Umleitung")

Ausnahmen:

- Allgemein bekannte Straßenabkürzungen wie "-str." für "-straße" sind zulässig
- Abkürzungen in Eigennamen bleiben erhalten (z.B. "Ph.-Rosenthal-Straße")

2. ZEITANGABE

2.1 Zeitraum-Struktur

- Zeitangabe MUSS an erster Stelle stehen (vor Linie und Maßnahmen)

Varianten:

- "ab [Datum]:"
- "vom [Datum] bis [Datum]:"
- "Ankündigung: [Datum] bis [Datum]:"
- Ansagetexte haben KEINE Zeitangabe

2.2 Datumsformat

- Konsistent verwenden ("Mo., 27. Oktober 2025")

3. MEHRERE LINIEN

3.1 Pluralisierung und Aufzählung

- **Gleicher Fahrzeugtyp:** Typ nur einmal → "BUS 72, 73 und N7 fahren..." (statt "BUS 72, BUS 73 und BUS N7")
- **Unterschiedlicher Fahrzeugtyp:** Jeden Typ einmal → "TRAM 2 und SEV15 fahren..."
- **Gemischte Typen:** "BUS 71, 73, 90 und TRAM 4, 7"

3.2 Gemeinsamer vs. separater Text

Gemeinsamer Text: Bei identischen/zusammenhängenden Maßnahmen (gleiche Route, gleiche Haltestellen) UND gleichem Zeitrahmen
Separater Text: Bei unterschiedlichen:

- Maßnahmen ODER
- Streckenabschnitten ODER
- **Zeitrahmen**

Beispiel separater Text bei unterschiedlichen Zeitrahmen:
• "vom 15. bis 20. November: BUS 72 fährt verkürzt bis und ab 'Hauptbahnhof'. Grund dafür sind Gleisbauarbeiten."
• "vom 18. bis 25. November: BUS 73 fährt verkürzt bis und ab 'Goerdelerring'. Grund dafür ist eine Straßensperrung."

4. SATZBAUSTEINE FÜR MASSNAHMEN

4.1 Verkürzung

- "fährt verkürzt bis und ab [Endstelle]"
- "fährt verkürzt nur bis und ab [Endstelle]"

4.2 Umleitung

Fahrzeugtyp-spezifisch:

- **TRAM/Zug:** "Die Züge werden [Beschreibung] umgeleitet"
- **BUS:** "Die Busse fahren eine Umleitung [Beschreibung]"

Standard:

- "mit Umleitung über [Straße1], [Straße2] und [Straße3]"
- "zwischen den Haltestellen [A] und [B] mit Umleitung über..."

Örtlich (kurze Strecke):

- "ab Haltestelle [A] mit örtlicher Umleitung zur Haltestelle [B]"

Optional:

- "Es kann zu Verspätungen kommen."

4.3 Haltestellenverlegung

Basis-Template: "Die Haltestelle '[Name]' ist [VERLEGUNG] verlegt."
[VERLEGUNG] - Varianten:

- **Einfach:** ∅ → "ist verlegt"
- **Mit Distanz:** "um [X] Meter [vor/hinter] den regulären Standort" / "vorverlegt" / "zurückverlegt"
- **Mit Adresse:** "in die [Straße] in Höhe Haus Nummer [Nr]"
- **Mit Orientierung:** "in Höhe [Orientierungspunkt]"
- **Richtungsspezifisch:** "Die Haltestelle '[Name]' ist in Fahrtrichtung '[Ziel1]' [VERLEGUNG1] verlegt. In Fahrtrichtung '[Ziel2]' [VERLEGUNG2] verlegt."

4.4 Ersatzhaltestelle

- "Die Haltestelle [Name] wird ersatzweise [Ortsbeschreibung] bedient"
- "Die Haltestelle [Name] wird ersatzweise auf der [Straße] in Höhe [Orientierungspunkt] bedient"

4.5 Halteausfall

Grundformel: "Die [Haltestelle/Halt] '[Name]' entfällt [in Fahrtrichtung '[Richtungstext]'] ersatzlos. Grund dafür ist [genaue Beschreibung des Grundes]."
Varianten:

- **Ein Ausfall:** "Die [Haltestelle/Halt] '[Name1]' entfällt ersatzlos."
- **Ein Ausfall in Fahrtrichtung:** "Die [Haltestelle/Halt] '[Name1]' entfällt in Fahrtrichtung '[Richtungstext]' ersatzlos."
- **Mehrere Ausfälle:** "Die [Haltestellen/Halte] '[Name1]', '[Name2]' und '[Name3]' entfallen ersatzlos."
- **Mit Ersatzhalt:** "Die [Haltestelle/Halt] '[Name]' entfällt. Ersatzweise kann [die Ersatzhaltestelle/der Ersatzhalt] '[Ersatzname]' genutzt werden."
- **Empfehlung:** "Wir empfehlen [die Haltestelle/den Halt] '[Name1]' zu benutzen."

4.6 Steigverschiebung

Singular:

- "An der Haltestelle '[Name]' ist der Steig [A/B/C] verschoben."

Plural:

- "An der Haltestelle '[Name]' sind die Steige [A] und [B] verschoben."

4.7 Geänderter Abfahrtsort

- "Geänderter Abfahrtsort: Die Linie [X] in Richtung [Ziel] verkehrt ab Steig [Y]."

4.8 Umsteigeempfehlung

Einfach:
- "Bitte nutzen Sie als Ersatz ab [Ort] die [Linie]."

Mit Umstieg:
- "Bitte fahren Sie ab [Ort] weiter bis [Ziel] und steigen dort in die Züge der Linie [X] um."

Umfahrung:
- "Umfahrungsmöglichkeit mit dem Bus [X] zwischen [A] und [B]."

4.9 Zeitabhängige Ausnahmen

- "Ausgenommen sind [Zeitspezifikation]."
- **Beispiel:** "Ausgenommen sind an Schultagen die Fahrten um 11:34 Uhr, 14:04 Uhr und 16:04 Uhr."
- **Position:** VOR dem Grund

4.10 Verstärkerverkehr (primär bei Events)

- "fahren mit zusätzlichen Straßenbahnen und Bussen"
- "verkehren im 10-Minuten-Takt"
- "im Anreise- und Abreisezeitraum"

4.11 Zusatzhalt

Grundformel: "Die Linie hält zusätzlich [in/an der] [Haltestelle/Halt] '[Name]'. Grund dafür ist [genaue Beschreibung des Grundes]."

Varianten:
- Ein Zusatzhalt
- **Mehrere Zusatzhalte:** "[in/an den] [Haltestellen/Halten] '[Name1]', '[Name2]' und '[Name3]'"

5. EVENTS (Sportveranstaltungen, Lichtfest, Messen)

5.1 Event-spezifische Bausteine

Fahrgastinformation:
- **Einleitung:** "Anlässlich des [Event-Name] ..." (steht nach Zeitangabe, vor Linie)
- **Eintrittskarte:** "Ihre Eintrittskarte gilt als Fahrtberechtigung im gesamten MDV-Gebiet."

Ansagetext:
- **Einleitung:** "anlässlich des [Event]" oder "wegen des [Event]"
- **Zeitangabe:** "heute" statt konkretes Datum
- **Eintrittskarte:** "Ihre Eintrittskarte gilt als Fahrtberechtigung."

6. ENDMELDUNGEN (Aufhebung von Einschränkungen)

6.1 Endmeldungs-Formulierungen

Grundformel: "Die Einschränkungen [an der Haltestelle/an dem Halt] '[Name]' sind aufgehoben."
Weitere Formulierungen:

- **Nach Verlegung:** "Die reguläre Haltestelle '[Name]' ist wieder in Betrieb."
- **Nach Ausfall:** "Die Haltestelle '[Name]' ist wieder in Betrieb."
- **Steige:** "An der Haltestelle '[Name]' ist der Steig A wieder in Betrieb." / "sind die Steige A und B wieder in Betrieb."

6.2 Kein Grund bei Endmeldungen

- **NICHT verwenden:** "verkehrt wieder auf originalen Linienweg"

7. GRUND-FORMULIERUNG

7.1 Format und Position (Fahrgastinformation)

Format: "Grund dafür ist [spezifische Beschreibung]." / "Grund dafür sind [spezifische Beschreibung]."
Position: Immer als letzter Satz (eigener Satz)
Beispiele:

- "Grund dafür ist eine Sperrung der Straße." (Singular)
- "Grund dafür sind Gleisbauarbeiten." (Plural)

7.2 Ausnahmen und Sonderfälle

Bei Fahrgastinformationen:

- Entfällt bei Endmeldungen
- Kann bei Events entfallen (wenn Event am Anfang genannt wurde)

Bei Ansagetexten:

- Integriert mit "wegen [Grund]" direkt nach Begrüßung

8. FORMATIERUNGSREGELN

8.1 Aufzählungen

- Letztes Element mit "und" (nicht Komma)
- "A, B und C" (nicht "A, B, C")

8.2 Haltestellen

- IMMER in Anführungszeichen

- "Paunsdorf", "Edlichstraße"

8.3 Richtungsspezifische Meldungen

Grundstruktur:
[ZEITRAHMEN], [LINIE + FAHRZEUGTYP] in Richtung [Ziel A]: [MASSNAHMEN]. In Richtung [Ziel B]: [MASSNAHMEN]. [GRUND]

Formatierungsregeln:
- Erste Richtung: Vollständige Formulierung mit "in Richtung [Ziel]:"
- Weitere Richtungen: Verkürzt mit "In Richtung [Ziel]:"
- Grund einmalig am Ende für beide/alle Richtungen
- Doppelpunkt nach jeder Richtungsangabe

Beispiele:

Einfache richtungsspezifische Meldung:
"ab Mo., 15. November 2025: BUS 72 in Richtung 'Paunsdorf': Die Haltestelle 'Markt' entfällt ersatzlos. In Richtung 'Connewitz': Die Haltestelle 'Augustusplatz' entfällt ersatzlos. Grund dafür sind Bauarbeiten."

Verkürzung mit unterschiedlichen Endstellen:
"vom Mo., 20. November bis Fr., 24. November 2025: TRAM 11 in Richtung 'Schkeuditz': fährt verkürzt bis und ab 'Hauptbahnhof'. In Richtung 'Markkleeberg-West': fährt verkürzt bis und ab 'Connewitzer Kreuz'. Grund dafür ist eine Streckensperrung."

9. ANSAGETEXTE - DETAILLIERTE REGELN

9.1 Struktur

Pflichtstruktur: Sehr geehrte Fahrgäste, wegen [GRUND] [MASSNAHME]. [DETAILS].

9.2 Regeln

- **Begrüßung:** IMMER "Sehr geehrte Fahrgäste" am Anfang
- **Grund:** IMMER integriert mit "wegen [Grund]" direkt nach Begrüßung
- **Zeitangabe:** NIEMALS im Ansagetext
- **Liniennummer:** NIEMALS - stattdessen "dieser Bus" / "diese Straßenbahn"
- **Richtungsangabe:** "Richtung [Ziel]" (NICHT "in Fahrtrichtung [Ziel]")
- **Kompaktheit:** Nur die wichtigsten Informationen

9.4 Maßnahmen-Formulierungen für Ansagetexte

Umleitung (richtungsspezifisch):
"Sehr geehrte Fahrgäste, wegen [Grund] verkehrt dieser Bus/diese Straßenbahn Richtung [Ziel] mit Umleitung über [Straße1], [Straße2] und [Straße3]. Die Haltestelle[n] '[Name(n)]' kann/können nicht bedient werden."

Umleitung (beide Richtungen):
"Sehr geehrte Fahrgäste, wegen [Grund] verkehrt diese Linie in beiden Richtungen mit Umleitung über [Straße1], [Straße2] und [Straße3]. Die Haltestelle[n] '[Name(n)]' kann/können nicht bedient werden."

Verkürzung:
"Sehr geehrte Fahrgäste, wegen [Grund] verkehrt diese Linie verkürzt bis und ab Ersatzendstelle '[Name]'."

Schienenersatzverkehr:
"Sehr geehrte Fahrgäste, wegen [Grund] ist ein Schienenersatzverkehr eingerichtet. Die Busse fahren zwischen '[A]' und '[B]' über [Haltestellen-Liste]."

Haltestellenverlegung:
"Sehr geehrte Fahrgäste, wegen [Grund] ist die Haltestelle '[Name]' verlegt. Bitte nutzen Sie die Ersatzhaltestelle [Ortsbeschreibung]."

Halteausfall:
"Sehr geehrte Fahrgäste, wegen [Grund] kann die Haltestelle '[Name]' nicht bedient werden. Ersatzweise kann die Haltestelle '[Alternative]' genutzt werden."

Events - Verstärkerverkehr:
"Sehr geehrte Fahrgäste, anlässlich des [Event] fahren heute die Linien [X], [Y] und [Z] mit zusätzlichen Straßenbahnen und Bussen. Ihre Eintrittskarte gilt als Fahrtberechtigung."

Events - Umleitung:
"Sehr geehrte Fahrgäste, wegen des [Event-Name] verkehrt diese Linie heute ab [Uhrzeit] mit Umleitung [Beschreibung]. Bitte beachten Sie die Haltestellenaushänge."

Endmeldung:
"Sehr geehrte Fahrgäste, die Einschränkungen an der Haltestelle '[Name]' sind aufgehoben. Die reguläre Haltestelle ist wieder in Betrieb."

B Sperrung der Lützner Straße

Leipziger Verkehrsbetriebe (LVB) GmbH Leipzig, 24.09.2025
BSP – He

Anweisung 198/25
Sperrung der Lützner Straße

• Anweisung Nr. 100/24 – Sperrung der Jahnallee

Linie N2, operativer SEV

Für den Neubau der landwärtigen Bushaltestelle sowie einer Querungshilfe wird

vom **Montag, 13.10.2025** bis **Freitag, 17.10.2025**

die Lützner Straße zwischen Brünner Straße und Nelkenweg für den Kfz-Verkehr in beiden Richtungen gesperrt.

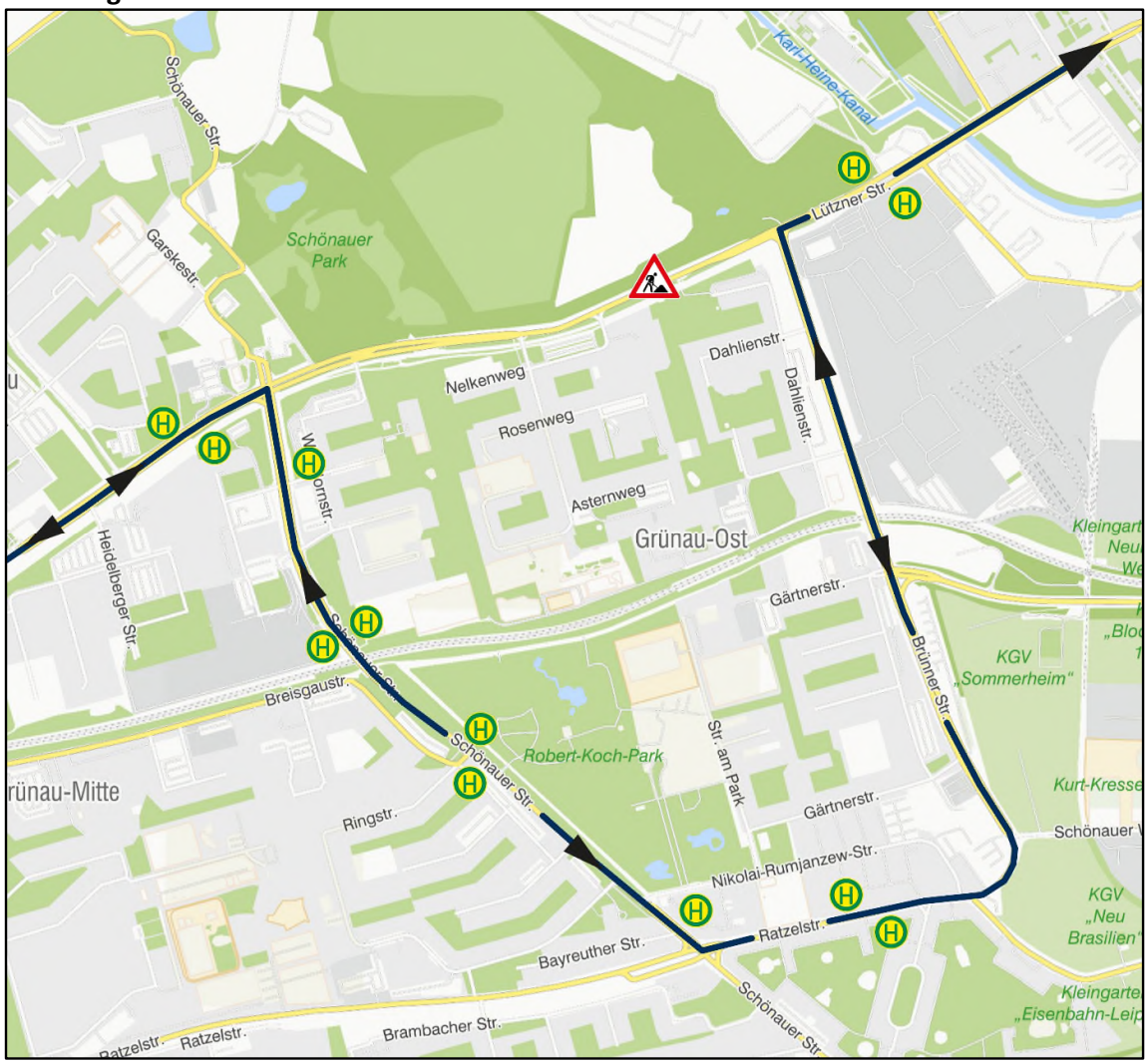
Linie N2 verkehrt ab der Nacht vom Mo. 13.10./ **Di. 14.10.** zwischen den Haltestellen Saarländer Straße und Schönauer Ring in **beiden Richtungen** mit Umleitung über Brünner Straße – Ratzelstraße – Schönauer Straße – Lützner Straße und bedient alle Haltestellen entlang der Umleitung.

Ansagetext Haltestelle Saarländer Straße -> Markranstädt:
Wegen Sperrung der Lützner Straße verkehrt diese Linie weiter mit Umleitung über Brünner Straße – Ratzelstraße – Schönauer Straße – Lützner Straße zur Haltestelle Schönauer Ring. Die Haltestellen Grünauer Allee und Parkallee können nicht bedient werden.

Ansagetext Haltestelle Schönauer Ring -> Hauptbahnhof:
Wegen Sperrung der Lützner Straße verkehrt diese Linie weiter mit Umleitung über Schönauer Straße – Ratzelstraße – Brünner Straße – Lützner Straße zur Haltestelle Saarländer Straße. Die Haltestellen Parkallee und Grünauer Allee können nicht bedient werden.

BMH
- Haltestellen Parkallee und Grünauer Allee in beiden Richtungen für die Linien N2 und X5 schließen,
- alle Haltestellen im Umleitungsweg der Linie N2 in beiden Richtungen einrichten,
- Fahrgastinformationen und Fahrpläne veröffentlichen.

Fahrtwegskizze:



BSP

C Gleisbauarbeiten in der Riesaer Straße

Leipziger Verkehrsbetriebe (LVB) GmbH Leipzig, 23.10.2025
BSP – Tr

Anweisung Nr. 200/25
Gleisbauarbeiten in der Riesaer Straße

- Anweisung Nr. 100/24 – Sperrung der Jahnallee
- Anweisung Nr. 130/25 – Leitungsbauarbeiten in der Zweinaundorfer Straße
- Anweisung Nr. 145/25 – Sperrung der Volksgartenstraße

Linien 2, 4, 7, 8, 9, 10, 12, 14, 16, 37, 72, 90, N17, SEV 7, SEV N17, Aus- und Einrücker

Im Zeitraum vom **Montag, 27.10.2025** bis **Freitag, 19.12.2025** werden Gleisbauarbeiten in der Riesaer Straße zwischen Ostheimstraße und Elisabeth-Schumacher-Straße sowie zwischen Barbarastraße und Karl-Bücher-Straße (Straßenbahnhof Paunsdorf) durchgeführt. Hierfür wird die Riesaer Straße zwischen Ostheimstraße und Elisabeth-Schumacher-Straße für den Verkehr in beiden Richtungen gesperrt. Zwischen Barbarastraße und Karl-Bücher-Straße (Straßenbahnhof Paunsdorf) wird die Riesaer Straße nur in stadteinwärtige Richtung gesperrt.

Linie 72 verlässt sofort nach Fahrgastausstieg die Ankunftshaltestelle Paunsdorf, Buswendestelle um dem Schienenersatzverkehr sowie der Linie 90 die Nutzung der Haltestelle zu gewährleisten.

Linie 90
- verkehrt mit geänderten Fahrplänen,
- verkehrt **in Richtung Wahren** ab Haltestelle Güterbahnhofstraße weiter über Riesaer Straße – rechts Busplatte, Bedienung der Ersatzhaltestelle Paunsdorf, Buswendestelle und weiter mit Umleitung über den Geh- und Radweg parallel zur Gleisanlage – links Permoser Straße zur Haltestelle Weinbrennerstraße und weiter in normaler Linienführung. Die neue Haltestelle Klettenstraße (Linie 79) auf der Permoserstraße wird nach der Inbetriebnahme mit bedient.
  Die Haltestellen Barbarastraße und Theodor-Heuss-Straße können nicht bedient werden,
- um eine Behinderung anderer Linien zu vermeiden, ist bei den an der Ankunftshaltestelle Paunsdorf, Buswendestelle (Linie 72) endenden Fahrten die Haltestelle sofort nach Bedienung zu verlassen.

Ansagetext Ersatzhaltestelle Paunsdorf, Buswendestelle -> Wahren:
…wegen Gleisbauarbeiten in der Riesaer Straße verkehrt diese Linie weiter mit Umleitung über die Permoser Straße zur Haltestelle Weinbrennerstraße.
Die Haltestellen Barbarastraße und Theodor-Heuss-Straße können nicht bedient werden.

BMH
- Ankunftshaltestelle Paunsdorf, Buswendestelle der Linie 72 für Linie 90 Richtung Wahren einrichten,
- Haltestelle Klettenstraße der Linie 79 Richtung Connewitz **(erst ab 24.11.2025)** für Linie 90 Richtung Wahren einrichten,
- alle Haltestellen zwischen Paunsdorf, Buswendestelle und Theodor-Heuss-Straße der Linie 90 Richtung Wahren schließen,
- Fahrgastinformationen und Fahrpläne veröffentlichen.

Straßenbahnsperrung:

Für die Gleisbauarbeiten wird

vom **Montag, 27.10.2025 4.00 Uhr** bis **Samstag, 06.12.2025 4.00 Uhr**

der Streckenabschnitt Wurzner Straße – Riesaer Straße zwischen Emmausstraße und Paunsdorf, Straßenbahnhof für den Straßenbahnverkehr in beiden Richtungen gesperrt.

Verkehrsorganisatorische Maßnahmen:

Aus- und Einrücker von/ zum Straßenbahnhof Paunsdorf verkehren über Torgauer Straße/ Paunsdorf-Nord und bedienen bei Fahrten über die Untere Torgauer Straße die Ersatzhaltestelle Edlichstraße in der Torgauer Straße Höhe Dornbergerstraße.

Linie 7 verkehrt bei allen Fahrten bis bzw. ab Paunsdorf, Straßenbahnhof sowie zwischen den Haltestellen Wiebelstraße und Straßenbahnhof Paunsdorf in beiden Richtungen über Torgauer Straße – Portitzer Allee – Heiterblickallee und bedient die Ersatzhaltestelle Edlichstraße in der Torgauer Straße Höhe Dornbergerstraße sowie alle Haltestellen entlang der Umleitung.

Linie 8 verkehrt nur bis bzw. ab Sellerhausen, Emmausstraße; Endstelle an der Haltestelle in der Schleife Emmausstraße, die Abfahrt erfolgt an der Haltestelle der Linie 7 in der Wurzner Straße.

Linie 37 verkehrt zwischen Paunsdorf, Straßenbahnhof und Sommerfeld.

Linie N17 verkehrt bei allen Fahrten bis bzw. ab Paunsdorf, Straßenbahnhof sowie zwischen den Haltestellen Wiebelstraße und Straßenbahnhof Paunsdorf in beiden Richtungen über Torgauer Straße – Portitzer Allee – Heiterblickallee und bedient die Ersatzhaltestelle Edlichstraße in der Torgauer Straße Höhe Dornbergerstraße sowie alle Haltestellen entlang der Umleitung.

SEV 7 wird zwischen Paunsdorf, Buswendestelle und Wiebelstraße eingerichtet und verkehrt in der Tagesrandlage über Torgauer Platz.

SEV N17 wird zwischen Paunsdorf, Buswendestelle und Torgauer Platz eingerichtet.

Es wird angewiesen:

Linien 2, 9, 10, 14, 16 Aus- und Einrücker (VH3) verkehren in beiden Richtungen über Torgauer Straße – Paunsdorf-Nord – Paunsdorf, Straßenbahnhof.

Linien 4, 12 Aus- und Einrücker (VH3) verkehren in beiden Richtungen über Torgauer Straße – Paunsdorf-Nord – Paunsdorf, Straßenbahnhof und bedienen die Ersatzhaltestelle Edlichstraße in der Torgauer Straße Höhe Dornbergerstraße.

Linie 7 verkehrt mit geänderten Fahrplänen.
Bedient zusätzlich in beide Richtungen die Ersatzhaltestelle Edlichstraße in der Torgauer Straße in Höhe Dornbergerstraße.
Bei den Fahrten in Richtung Paunsdorf, Straßenbahnhof sind Fahrgäste mit Ziel Sommerfeld an der Haltestelle Paunsdorf-Nord an die Linie 3 bzw. die Linie 37 zu verweisen. An der Haltestelle Paunsdorf, Straßenbahnhof ist auf ankommende Busse des Schienenersatzverkehrs zu achten und den Fahrgästen der Umstieg zu ermöglichen.
Bei den Fahrten in Richtung Böhlitz-Ehrenberg sind Fahrgäste mit Ziel Sommerfeld an der Haltestelle Paunsdorf-Nord an die Linie 3E bzw. die Linie 37 zu verweisen.

Beschilderung:

ab Paunsdorf, Straßenbahnhof:	Ziel 80 „Böhlitz-Ehrenberg über Torgauer Platz“
ab Böhlitz-Ehrenberg:	Ziel 81 „Paunsdorf Strbf. über Torgauer Platz.“

Ansagetext Wiebelstraße -> Paunsdorf, Straßenbahnhof:
…wegen Gleisbauarbeiten in der Riesaer Straße verkehrt diese Linie weiter mit Umleitung über Torgauer Straße – Portitzer Allee – Paunsdorf-Nord zur Ersatzendstelle Paunsdorf, Straßenbahnhof. Die Haltestellen Hermelinstraße bis Sommerfeld erreichen Sie mit der Linie 3E bzw. Linie 37 ab der Haltestelle Paunsdorf-Nord. Die Haltestellen Annenstraße bis Barbarastraße erreichen Sie mit dem Schienenersatzverkehr ab dieser Haltestelle.

Ansagetext Paunsdorf-Nord -> Paunsdorf, Straßenbahnhof:
…die Haltestellen Hermelinstraße, Paunsdorf Center und Sommerfeld erreichen Sie mit den Linien 3E bzw. 37 ab der Haltestelle Paunsdorf-Nord.

Ansagetext Paunsdorf, Straßenbahnhof:
…diese Linie endet hier. Die Haltestellen Theodor-Heuss-Straße bis Wiebelstraße erreichen Sie mit dem Schienenersatzverkehr ab der Ersatzhaltestelle Paunsdorf, Buswendestelle. Die Haltestelle Barbarastraße kann nicht bedient werden.

Ansagetext Paunsdorf-Nord -> Böhlitz-Ehrenberg:
…die Haltestellen Hermelinstraße, Paunsdorf Center und Sommerfeld erreichen Sie mit den Linien 3E bzw. 37 ab der Haltestelle Paunsdorf-Nord.

Linie 8 verkehrt mit geänderten Fahrplänen.

Beschilderung:

ab Grünau-Nord:	Ziel 46	„Sellerhausen“

Ansagetext Sellerhausen, Emmausstraße -> Paunsdorf-Nord:

…diese Linie endet hier. Die Haltestellen Ostheimstraße bis Paunsdorf, Straßenbahnhof erreichen Sie mit dem Schienenersatzverkehr ab Haltestelle Sellerhausen, Emmausstraße der Linie 7 in der Wurzner Straße.

Linie 37 verkehrt nach Fahrplan.

Beschilderung:

ab Paunsdorf, Straßenbahnhof:	Ziel 16	„Sommerfeld“
ab Sommerfeld:	Ziel 48	„Paunsdorf, Straßenbahnhof“

Ansagetext Paunsdorf-Nord -> Paunsdorf, Straßenbahnhof:

…Fahrgäste mit Ziel Reudnitz nutzen bitte die umgeleitete Linie 7 ab der Haltestelle Paunsdorf-Nord.

Ansagetext Paunsdorf, Straßenbahnhof:

…diese Linie endet hier. Die Haltestellen Theodor-Heuss-Straße bis Wiebelstraße erreichen Sie mit dem Schienenersatzverkehr ab der Ersatzhaltestelle Paunsdorf, Buswendestelle. Die Haltestelle Barbarastraße kann nicht bedient werden.

Linie N17 verkehrt mit geänderten Fahrplänen.
Bedient zusätzlich in beide Richtungen die Ersatzhaltestelle Edlichstraße in der Torgauer Straße in Höhe Dornbergerstraße.
An der Haltestelle Paunsdorf, Straßenbahnhof bzw. Torgauer Platz ist auf ankommende Busse des Schienenersatzverkehrs zu achten und den Fahrgästen der Umstieg zu ermöglichen.

Beschilderung:

ab Paunsdorf, Straßenbahnhof:	Ziel 138 „Lausen über Hauptbahnhof“
ab Lausen:	Ziel 81 „Paunsdorf Strbf. über Torgauer Platz.“

Ansagetext Haltestelle Torgauer Platz -> Paunsdorf, Straßenbahnhof:

…wegen Gleisbauarbeiten in der Riesaer Straße verkehrt diese Linie weiter mit Umleitung über Torgauer Straße – Portitzer Allee – Paunsdorf-Nord zur Ersatzendstelle Paunsdorf, Straßenbahnhof. Die Haltestellen Geißlerstraße, Bülowviertel bis Barbarastraße erreichen Sie mit dem Schienenersatzverkehr ab dieser Haltestelle.

Ansagetext Paunsdorf, Straßenbahnhof:

…diese Linie endet hier. Die Haltestellen Theodor-Heuss-Straße bis Edlichstraße erreichen Sie mit dem Schienenersatzverkehr ab der Ersatzhaltestelle Paunsdorf, Buswendestelle. Die Haltestelle Barbarastraße kann nicht bedient werden.

SEV 7

• verkehrt nach Fahrplan,
• verlässt sofort nach Fahrgastausstieg die Ankunftshaltestelle Paunsdorf, Buswendestelle um dem Schienenersatzverkehr sowie der Linie 90 die Nutzung der Haltestelle zu gewährleisten.

Fahrtweg: Paunsdorf, Buswendestelle – Rad- und Gehweg – links Permoserstraße – links Theodor-Heuss-Straße – rechts Geithainer Straße – rechts Watzdorfstraße – links Wurzner Straße – rechts Lilienstraße.
= 15 Minuten Fahrtzeit

zurück: Dresdner Straße – links Wurzner Straße – rechts Watzdorfstraße – links Geithainer Straße – links Theodor-Heuss-Straße – rechts Riesaer Straße – links Paunsdorf, Buswendestelle. **= 16 Minuten Fahrtzeit**

wenden: Lilienstraße – links Kohlgartenstraße – links Dresdner Straße.

Haltestellenbedienung:

Paunsdorf, Buswendestelle	Ersatzhaltestelle vor HST. Linie 72
Klettenstraße stw.	Bus 79 **(erst ab 24.11.2025)**
Theodor-Heuss-Straße stw.	Bus 79
Bf. Paunsdorf stw.	Bus 79
Ostheimstraße stw.	Ersatzhaltestelle in Watzdorfstraße
Sellerhausen, Emmausstraße stw.	Bus 77
Annenstraße stw.	Strab
Edlichstraße stw.	Strab
Wiebelstraße stw.	Strab
Lilienstraße	Ersatzhaltestelle nur Bereitstellung
Reudnitz, Koehlerstraße ldw.	Bus 70
Wiebelstraße ldw.	Strab
Edlichstraße ldw.	Strab
Annenstraße ldw.	Strab
Sellerhausen, Emmausstraße ldw.	Bus 77
Ostheimstraße ldw.	Ersatzhaltestelle in Wurzner Straße
Bf. Paunsdorf ldw.	Bus 79
Barbarastraße ldw.	Bus 90
Paunsdorf, Straßenbahnhof ldw.	Bus 90
Paunsdorf, Buswendestelle	Linie 72 Ankunft

Beschilderung:

ab Paunsdorf, Straßenbahnhof:	Ziel 246 „Wiebelstraße“
ab Reudnitz, Koehlerstraße:	Ziel 48 „Paunsdorf, Straßenbahnhof“

Ansagetext Paunsdorf, Buswendestelle Ankunft:

…der Schienenersatzverkehr endet an dieser Haltestelle. Bitte nutzen Sie zur Weiterfahrt in Richtung Sommerfeld die Straßenbahnlinie 37 sowie in Richtung Böhlitz-Ehrenberg die Linie 7.

Ansagetext Wiebelstraße -> Reudnitz:
…der Schienenersatzverkehr endet an dieser Haltestelle. Bitte nutzen Sie zur Weiterfahrt die Straßenbahn ab dieser Haltestelle.

SEV7 über Torgauer Platz • verkehrt nach Fahrplan,
• in **Richtung Wiebelstraße** ist nach Bedienung der Haltestelle Torgauer Platz (Linie 8 -> Grünau) bis zur Haltlinie vorzufahren und am LZA-Mast mittels Schlüsselschalter eine Freigabe des ÖPNV-Signal nach links anzufordern,
• verlässt sofort nach Fahrgastausstieg die Ankunftshaltestelle Paunsdorf, Buswendestelle um dem Schienenersatzverkehr sowie der Linie 90 die Nutzung der Haltestelle zu gewährleisten.

Fahrtweg: Paunsdorf, Buswendestelle – Rad- und Gehweg – links Permoserstraße – links Theodor-Heuss-Straße – rechts Geithainer Straße – rechts Watzdorfstraße – links Wurzner Straße – rechts Annenstraße – links Eisenbahnstraße – links Torgauer – Wurzner Straße – rechts Lilienstraße.
= 16 Minuten Fahrtzeit

zurück: Dresdner Straße – links Wurzner Straße – Torgauer Straße – rechts Eisenbahnstraße – rechts Annenstraße – links Wurzner Straße – rechts Watzdorfstraße – links Geithainer Straße – links Theodor-Heuss-Straße – rechts Riesaer Straße – links Paunsdorf, Buswendestelle. **= 16 Minuten Fahrtzeit**

wenden: Lilienstraße – links Kohlgartenstraße – links Dresdner Straße.

Haltestellenbedienung:

Paunsdorf, Buswendestelle	Ersatzhaltestelle vor HST. Linie 72
Klettenstraße stw.	Bus 79 **(erst ab 24.11.2025)**
Theodor-Heuss-Straße stw.	Bus 79
Bf. Paunsdorf stw.	Bus 79
Ostheimstraße stw.	Ersatzhaltestelle in Watzdorfstraße
Sellerhausen, Emmausstraße stw.	Bus 77
Annenstraße stw.	Strab
Geißlerstraße, Bülowviertel stw.	Strab
Torgauer Platz stw.	Strab
Edlichstraße stw.	Ersatzhaltestelle Torgauer Straße
Wiebelstraße stw.	Strab
Lilienstraße	Ersatzhaltestelle nur Bereitstellung
Reudnitz, Koehlerstraße ldw.	Bus 70
Wiebelstraße ldw.	Strab
Edlichstraße ldw.	Ersatzhaltestelle Torgauer Straße
Torgauer Platz ldw.	Strab
Geißlerstraße, Bülowviertel ldw.	Strab
Annenstraße ldw.	Strab
Sellerhausen, Emmausstraße ldw.	Bus 77

Ostheimstraße ldw.	Ersatzhaltestelle in Wurzner Straße
Bf. Paunsdorf ldw.	Bus 79
Barbarastraße ldw.	Bus 90
Paunsdorf, Straßenbahnhof ldw.	Bus 90
Paunsdorf, Buswendestelle	Linie 72 Ankunft

Beschilderung:

ab Paunsdorf, Straßenbahnhof:	Ziel 246 „Wiebelstraße“
ab Reudnitz, Koehlerstraße:	Ziel 81 „Paunsdorf Strbf. über Torgauer Platz.“

Ansagetext Paunsdorf, Buswendestelle Ankunft:

…der Schienenersatzverkehr endet an dieser Haltestelle. Bitte nutzen Sie zur Weiterfahrt in Richtung Sommerfeld die Straßenbahnlinie 37 sowie in Richtung Böhlitz-Ehrenberg die Linie 7.

Ansagetext Wiebelstraße -> Reudnitz:

…der Schienenersatzverkehr endet an dieser Haltestelle. Bitte nutzen Sie zur Weiterfahrt die Straßenbahn ab dieser Haltestelle.

SEV N17 verkehrt nach Fahrplan.

Fahrtweg: Paunsdorf, Buswendestelle – Rad- und Gehweg – links Permoserstraße – links Theodor-Heuss-Straße – rechts Geithainer Straße – rechts Watzdorfstraße – links Wurzner Straße – rechts Torgauer Straße.
= 13 Minuten Fahrtzeit

zurück: Torgauer Straße – rechts Eisenbahnstraße – rechts Annenstraße – links Wurzner Straße – rechts Watzdorfstraße – links Geithainer Straße – links Theodor-Heuss-Straße – rechts Riesaer Straße – links Paunsdorf, Buswendestelle.
= 10 Minuten Fahrtzeit

Haltestellenbedienung:

Paunsdorf, Buswendestelle	Ersatzhaltestelle vor HST. Linie 72
Klettenstraße stw.	Bus 79 **(erst ab 24.11.2025)**
Theodor-Heuss-Straße stw.	Bus 79
Bf. Paunsdorf stw.	Bus 79
Ostheimstraße stw.	Ersatzhaltestelle in Watzdorfstraße
Sellerhausen, Emmausstraße stw.	Bus 77
Annenstraße stw.	Strab
Edlichstraße stw.	Strab
Edlichstraße ldw.	Ersatzhaltestelle Torgauer Straße
Torgauer Platz ldw.	Strab (hier Anschluss abwarten)
Torgauer Platz ldw.	Strab
Geißlerstraße, Bülowviertel ldw.	Strab
Annenstraße ldw.	Strab
Sellerhausen, Emmausstraße ldw.	Bus 77

Ostheimstraße ldw.	Ersatzhaltestelle in Wurzner Straße
Bf. Paunsdorf ldw.	Bus 79
Barbarastraße ldw.	Bus 90
Paunsdorf, Straßenbahnhof ldw.	Bus 90
Paunsdorf, Buswendestelle	Linie 72 Ankunft

Beschilderung:

ab Paunsdorf, Straßenbahnhof:	Ziel 379 „Torgauer Platz“
ab Torgauer Platz:	Ziel 48 „Paunsdorf, Straßenbahnhof“

Ansagetext Paunsdorf, Buswendestelle Ankunft:

…der Schienenersatzverkehr endet an dieser Haltestelle. Bitte nutzen Sie zur Weiterfahrt in Richtung Paunsdorf-Nord die Straßenbahnlinie N17.

Ansagetext Ersatzhaltestelle Edlichstraße -> Torgauer Platz:

…der Schienenersatzverkehr endet am Torgauer Platz. Bitte nutzen Sie zur Weiterfahrt in Richtung Hauptbahnhof die Straßenbahn N17 ab der Ersatzhaltestelle auf der gegenüberliegenden Straßenseite.

BMH

- eine Ersatzhaltestelle Paunsdorf, Buswendestelle auf der Busplatte vor der Haltestelle Linie 72 für Abfahrt SEV einrichten,
- Haltestelle Klettenstraße der Linie 79 Richtung Connewitz **(erst ab 24.11.2025)** für SEV Richtung Wiebelstraße einrichten,
- alle Haltestellen entlang der Umleitung für Linie 7 und N17 einrichten,
- alle Haltestellen für Linie 7 und N17 zw. Edlichstraße und Barbarastraße sowie die Haltestellen Annenstraße und Geißlerstraße, Bülowviertel in beide Richtungen schließen,
- alle Haltestellen für Linie 7 zw. Hermelinstraße und Sommerfeld in beide Richtungen schließen,
- Haltestelle Torgauer Platz in der Eisenbahnstraße für Linie 7 und N17 schließen,
- alle Haltestellen der Linie 8 zw. Ostheimstraße und Paunsdorf-Nord in beide Richtungen schließen,
- Haltestelle Sellerhausen, Emmausstraße für Linie 8 Richtung Paunsdorf-Nord schließen,
- alle Haltestellen entlang der Linienführung der Linie 37 einrichten,
- eine Ersatzhaltestelle Edlichstraße in der Torgauer Straße in Höhe Dornbergerstraße für die Linien 7, N17 und SEV in beide Richtungen einrichten,
- alle Haltestellen entlang des Fahrtweges des SEV einrichten,
- eine Ersatzhaltestelle Ostheimstraße in der Wurzner Straße bzw. in der Watzdorfstraße für SEV einrichten,
- Fahrgastinformationen und Fahrpläne veröffentlichen.

BISF Weichen abschalten und Signale anbringen:
• EW-Nr. 470 (Wurzner/ Cunnersdorfer Straße ldw.) – W15/ W12

Sh 2 anbringen:
• Wurzner Straße ldw. (EW 470) nach Rechtsabzweig zur Cunnersdorfer Straße,
• Emmausstraße (HW 471) nach Linksabzweig zur Wurzner Straße stw.,
• Riesaer Straße stw. Richtung Karl-Bücher-Straße bis an Absperrung bzw. 50 Meter nach Ausfahrt VH3 (Gleis wird bis dorthin zum Rangieren benötigt!).

BSV-a E-Weichen festlegen:
• EW-Nr. 470 nach „Rechts“

Handweiche festlegen
• Nr. 471 nach „Links“

BFSF Während der Einsatzzeit der Linien 7, 37 und N17 muss die Durchfahrt durch den Straßenbahnhof Paunsdorf immer gewährleistet sein. Das Gleis 30 darf nicht durch einrückende Fahrzeuge blockiert werden. Ebenso muss die Durchfahrt des SEV 7 und SEV N17 auf der Buswendestelle immer gewährleistet sein.

Durchfahrtsgleise in VH3:
Gleis 4 -> Linie 7,
Gleis 5 -> Linie 37,
Gleis 9 -> Linie N17.

BMK-v prüft das Lichtraumprofil und stellt dieses gegebenenfalls her.

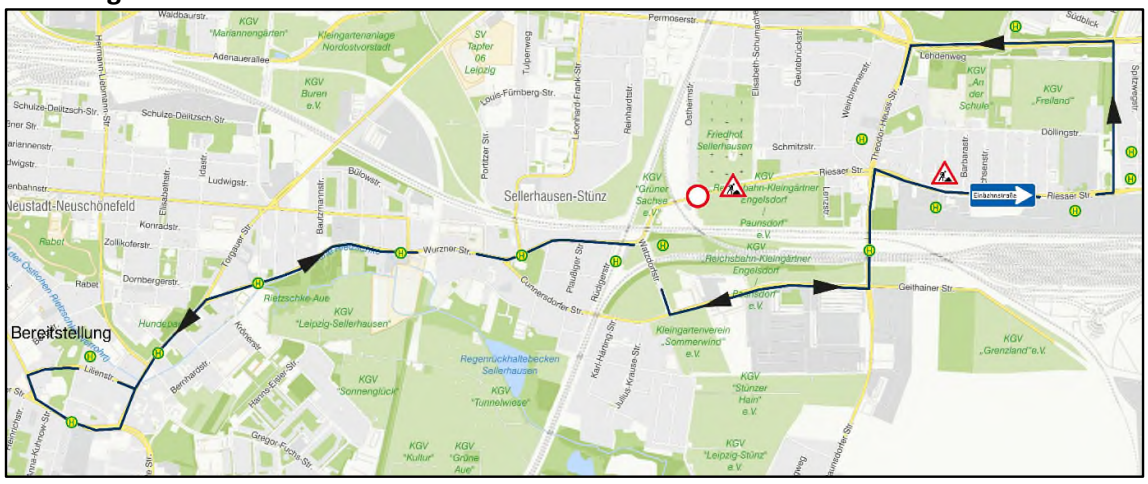
BMK-w stellt gegebenenfalls den Winterdienst auf dem Geh- und Radweg sicher.

BMG-k stellt die Nutzung der Toilette (im Imbiss) an der Wendestelle Sellerhausen, Emmausstraße sicher.

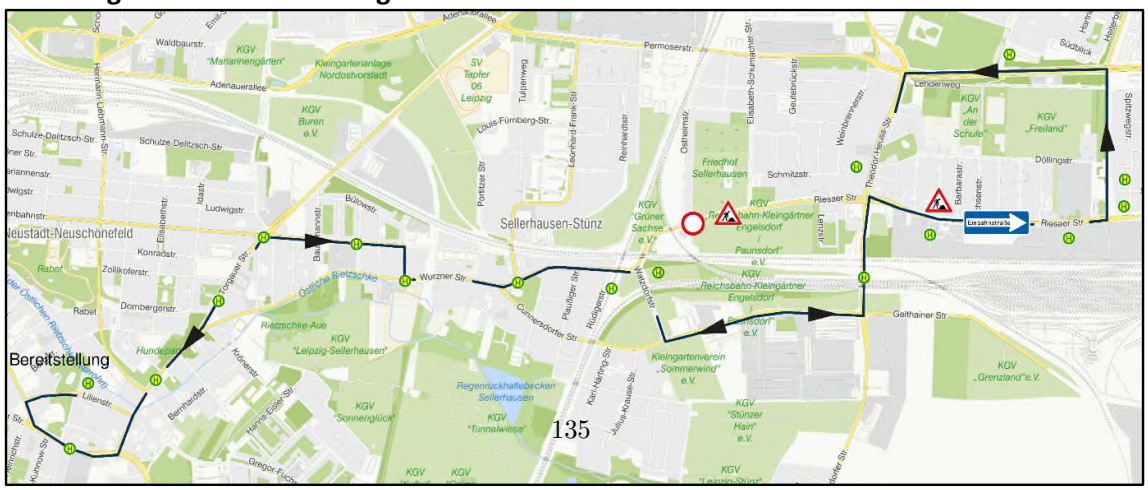
MMPV-v LZA-Beeinflussung entlang der Umleitungen aller betroffenen Linien anpassen.

Fahrtwegskizze Linie 90:

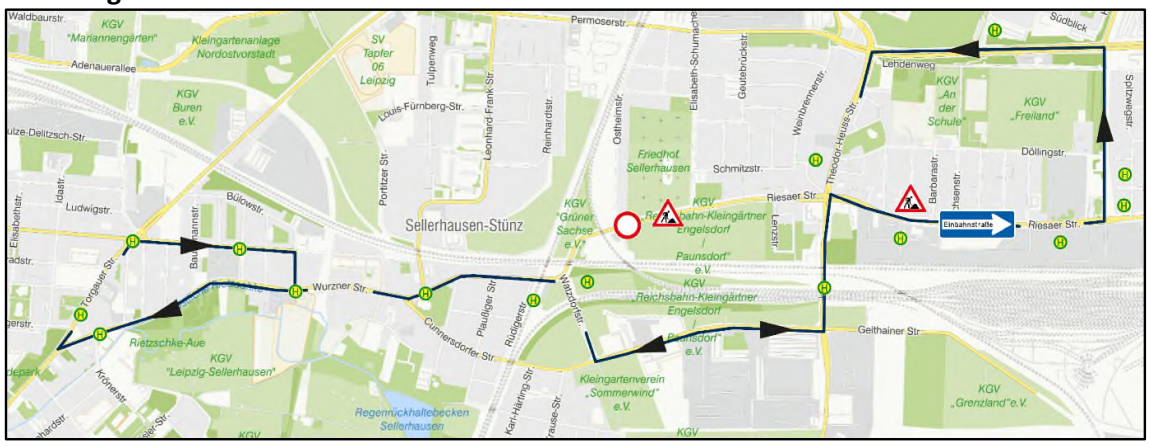
Fahrtwegskizze SEV7:



Fahrtwegskizze SEV7 über Torgauer Platz:



Fahrtwegskizze SEV N17:



Als allererstes, hier noch die letzte Referenz zum folgenden Listing im Anhang: C.1 Die hier benutzte Arbeit entstand im Rahmen des Moduls *Wissenschaftliches Arbeiten* aus dem 4. Semester. Da keine Bilder oder Grafiken vorgesehen waren, habe ich ein paar eher weniger relevante eingefügt. Trotzdem hoffe ich, dass der Leseflow der Arbeit einigermaßen angenehm ist. Alle für diese Abgabe relevanten Inhalte sind gleichmäßig über die Arbeit verteilt und können über die Seite 'Hinweise' angewählt werden. Überflüssige stellen der Arbeit wurden gelöscht, um den Inhalt bündig beisammen zu halten und nicht zu dünn zu verstreuen. Deshalb könnte der Text bei genauerem lesen allerdings auch teilweise nur wenig Sinn ergeben.

```
5
6         /*
7         Java Hello World example.
8         */
9
10        public class HelloWorldExample{
11
12          public static void main(String args[]){
13
14            /*
15            Use System.out.println() to print on console.
16            */
17            System.out.println("Hello World !");
18
19          }
20
21        }
22
23        /*
24        OUTPUT of the above given Java Hello World Example would be :
25        Hello World !
26        */
27
```

Listing C.1: Every java dev says 'Hello world!', but noone ever says 'How are you doing, world?'...

Hier noch die letzte Anforderung an das Dokument in Form eines Bildes von unserem Hochschullogo im Anhang.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abbildung C.1: Das Logo unserer Hochschule

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und dabei keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wernigerode, den 18.08.1999

Philipp Thüring