



METHODIK EINES INGENIEURS

Vorgelegt von:

Philipp Thüring
m26665
Friedrichstraße 12
38855 Wernigerode

Erstprüfer	Johr	Alexander
Zweitprüfer	Singer	Jürgen
Abgabedatum	28. Februar	2021

Inhaltsverzeichnis

1 Einleitung	5
1.1 Motivation und Problemstellung	5
1.2 Zielsetzung und Forschungsfragen	5
1.3 Wissenschaftlicher Beitrag und Forschungslücke	5
1.4 Aufbau der Arbeit	5
2 Theoretische Grundlagen und Stand der Forschung	6
2.1 Natural Language Processing für Verkehrsinformationen	6
2.1.1 Grundlagen der Sprachverarbeitung	6
2.1.2 NLP-Anwendungen im Verkehrssektor	10
2.1.3 Domänenspezifische Sprachmodelle	10
2.2 Sprachmodelle und Fine-Tuning	10
2.2.1 Fine-Tuning-Methoden	10
2.2.2 Prompt Engineering	10
2.2.3 Datensätze für Fine-Tuning	10
2.2.4 Herausforderungen beim Fine-Tuning	10
2.3 Ressourceneffizienz und Modelloptimierung	10
2.3.1 Problematik großer Sprachmodelle	10
2.3.2 Quantisierung	10
2.3.3 Weitere Optimierungsansätze	10
2.4 Qualitätssicherung bei KI-generierten Texten	10
2.4.1 Evaluationsmetriken für NLP	10
2.4.2 Halluzination Detection und Validierung	10
3 Anwendungskontext und Anforderungen	11
3.1 Leipziger Verkehrsbetriebe	11
3.2 Analyse der Verkehrsanweisungen	11
3.3 Anforderungen an das System	11
3.3.1 Funktionale Anforderungen	11
3.3.2 Nicht-funktionale Anforderungen	11
4 Modellauswahl: leo-mistral-hessianai-7b	12
4.1 Anforderungen an das Modell	12
4.2 Modellvergleich und Auswahlprozess	12
4.3 Modellvarianten für Deployment	12
5 Datensatzerstellung und -aufbereitung	13
5.1 Datenerhebung	13
5.2 Datensatzentwicklung für Fine-Tuning	13
5.2.1 Annotationsstrategie	13
5.2.2 Datenaugmentierung	13
5.2.3 Datensatzstruktur	13
5.2.4 Herausforderungen bei kleinem Datensatz	13

6 Implementierung	14
6.1 Technische Umsetzung	14
6.2 Fine-Tuning des Modells	14
6.2.1 LoRA-basiertes Fine-Tuning	14
6.2.2 Prompt Engineering und Automatisierung	14
6.2.3 Quantisierung	14
6.3 Qualitätssicherung und Validierung	14
7 Evaluation	15
7.1 Evaluationsmethodik	15
7.2 Qualitätsevaluation	15
7.2.1 Vollmodell vs. Quantisierte Versionen	15
7.2.2 Automatisierung und Konsistenz	15
7.3 Gesamtbewertung	15
8 Diskussion	16
8.1 Interpretation der Ergebnisse	16
8.2 Kritische Bewertung	16
8.2.1 Limitationen der Arbeit	16
8.2.2 Validität der Ergebnisse	16
9 Zusammenfassung und Ausblick	17
9.1 Zusammenfassung der Arbeit	17
9.2 Beantwortung der Forschungsfragen	17
9.3 Wissenschaftlicher und praktischer Beitrag	17
9.4 Ausblick und zukünftige Forschung	17
9.4.1 Praktische Implikationen	17
9.4.2 Erweiterungsmöglichkeiten	17
9.4.3 Offene Forschungsfragen	17
Hinweise zur Abgabe	21
10 Was ist ein Problem?	23
10.1 Veränderung	23
10.1.1 Am Besten	23
10.1.2 Ungewissheit	24
10.2 Der Kern des Ingenieursproblems	24
11 Analyse	25
11.1 Beschreibung des Problems	25
11.2 Performance Kriterien festlegen	26
11.3 Themenverwandte Arbeiten untersuchen	27
11.4 Ziel formulieren	28

12 Synthese	29
12.1 Lösungsansätze implementieren	29
12.1.1 Experimente Designen	30
12.1.2 Experimente durchführen	32
12.1.3 Ergebnisse extrahieren	32
Anhang	34
A Nachbereitung	34
Eidesstattliche Erklärung	37

1 Einleitung

1.1 Motivation und Problemstellung

1.2 Zielsetzung und Forschungsfragen

1.3 Wissenschaftlicher Beitrag und Forschungslücke

1.4 Aufbau der Arbeit

2 Theoretische Grundlagen und Stand der Forschung

2.1 Natural Language Processing für Verkehrsinformationen

2.1.1 Grundlagen der Sprachverarbeitung

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

Transformer-Architektur Die Transformer-Architektur wurde 2017 von Vaswani et al. mit dem wegweisenden Paper „Attention is All You Need“ eingeführt. Vaswani u. a., „Attention Is All You Need“. Im Gegensatz zu vorherigen Ansätzen wie Recurrent Neural Networks (RNNs) oder Long Short-Term Memory (LSTM) verzichtet die Transformer-Architektur vollständig auf rekurrente Strukturen und basiert stattdessen auf dem Attention-Mechanismus. Dieser fundamentale Paradigmenwechsel ermöglicht die parallele Verarbeitung von Sequenzen und führt zu deutlich schnelleren Trainingszeiten sowie besserer Skalierbarkeit.

Das Kernprinzip der Transformer-Architektur ist der Self-Attention-Mechanismus Vaswani u. a., „Attention Is All You Need“. Dieser ermöglicht es jedem Token in einer Sequenz, auf alle anderen Tokens im Kontext zuzugreifen und deren Relevanz für die eigene Repräsentation zu bewerten. Durch den Einsatz von Multi-Head Attention werden mehrere parallele Attention-Mechanismen verwendet, die unterschiedliche Aspekte der Kontextbeziehungen erfassen können Vaswani u. a., „Attention Is All You Need“. Diese Architektur ermöglicht es dem Modell, komplexe syntaktische und semantische Abhängigkeiten auch über große Distanzen im Text hinweg zu modellieren, ohne unter dem Vanishing-Gradient-Problem zu leiden, das RNNs bei langen Sequenzen beeinträchtigt.

Transformer-basierte Modelle lassen sich in drei Hauptkategorien einteilen: Encoder-Only-Modelle wie BERT Devlin u. a., „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“, die primär für bidirektionale Textklassifikation und Embeddings konzipiert sind, Decoder-Only-Modelle wie die GPT-Serie Radford, Narasimhan u. a., „Improving Language Understanding by Generative Pre-Training“; Radford, Wu u. a., „Language Models are Unsupervised Multitask Learners“, die für autoregressive Textgenerierung optimiert

sind, sowie Encoder-Decoder-Architekturen wie der ursprüngliche Transformer Vaswani u. a., „Attention Is All You Need“ und T5 Raffel u. a., „Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer“, die vor allem für Übersetzungsaufgaben entwickelt wurden. Für die in dieser Arbeit behandelte Aufgabe der Textgenerierung und -transformation ist die Decoder-Only-Architektur besonders geeignet, da sie speziell für die sequenzielle Erzeugung von Text konzipiert wurde.

Vortrainierte Sprachmodelle Ein zentrales Konzept moderner NLP-Systeme ist das Pretraining von Sprachmodellen auf großen, unlabeled Textkorpora. Durch Self-Supervised Learning, bei dem das Modell darauf trainiert wird, das jeweils nächste Token in einer Sequenz vorherzusagen Radford, Narasimhan u. a., „Improving Language Understanding by Generative Pre-Training“, entwickeln diese Modelle ein umfassendes Sprachverständnis inklusive Syntax, Semantik und implizitem Weltwissen. Zu den einflussreichsten vortrainierten Modellen gehören BERT Devlin u. a., „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“ mit seinem bidirektionalen Masked Language Modeling-Ansatz, die GPT-Serie Radford, Narasimhan u. a., „Improving Language Understanding by Generative Pre-Training“; Radford, Wu u. a., „Language Models are Unsupervised Multitask Learners“ für autoregressive Textgenerierung sowie T5 Raffel u. a., „Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer“ mit seinem universellen Text-to-Text-Framework.

Für die vorliegende Arbeit ist insbesondere die Mistral-7B-Architektur von Bedeutung, da sie die Grundlage für das verwendete LeoLM-Modell bildet. Mistral 7B Jiang u. a., „Mistral 7B“ ist ein hocheffizienter Decoder-Only-Transformer mit 7 Milliarden Parametern, der mehrere innovative Architekturmerkmale aufweist. Grouped-Query Attention (GQA) reduziert die Größe des Key-Value-Cache und ermöglicht schnellere Inferenz Jiang u. a., „Mistral 7B“. Sliding Window Attention erlaubt die effiziente Verarbeitung langer Kontexte, während der Rolling Buffer Cache die Speichernutzung optimiert Jiang u. a., „Mistral 7B“. Mit 7 Milliarden Parametern stellt Mistral einen Sweet Spot zwischen Modellleistung und Ressourceneffizienz dar und übertrifft in Benchmarks viele deutlich größere Modelle Jiang u. a., „Mistral 7B“. Als Open-Source-Modell unter Apache 2.0 Lizenz ist es besonders für lokale Ausführung und Fine-Tuning geeignet.

Ein wichtiger Unterschied besteht zwischen Base Models und Instruct Models. Base Models wie Mistral-7B Jiang u. a., „Mistral 7B“ sind auf reines Language Modeling trainiert Radford, Wu u. a., „Language Models are Unsupervised Multitask Learners“ und setzen primär Texte fort, ohne notwendigerweise expliziten Anweisungen zu folgen. Sie sind als Ausgangspunkt für aufgabenspezifisches Fine-Tuning konzipiert. Instruct Models hingegen durchlaufen zusätzlich ein Instruction Tuning Wei u. a., „Finetuned Language Models Are Zero-Shot Learners“, bei dem sie mittels Supervised Fine-Tuning auf Instruktionsdatensätzen

trainiert werden, um gezielt Anweisungen zu befolgen. Optional kann dieser Prozess durch Reinforcement Learning from Human Feedback (RLHF) Ouyang u. a., „Training Language Models to Follow Instructions with Human Feedback“ weiter verfeinert werden. Mistral bietet beide Varianten an: Mistral-7B als Base Model und Mistral-7B-Instruct als instruction-tuned Version Jiang u. a., „Mistral 7B“. Der detaillierte Vergleich und die Auswahlbegründung zwischen diesen Varianten erfolgt in Kapitel 4.

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 HessianAI, *LeoLM: Linguistically Enhanced Open Language Model for German*. Das in dieser Arbeit verwendete Modell leo-mistral-hessianai-7b ist eine Base-Variante, die speziell für die deutsche Sprache optimiert wurde HessianAI, *LeoLM: Linguistically Enhanced Open Language Model for German*. Diese Adaption kombiniert die architektonischen Vorteile und die Effizienz von Mistral mit erhöhter Sprachkompetenz im Deutschen und erzielt dadurch bessere Ergebnisse für deutschsprachige Anwendungen als rein englischsprachige Basismodelle. Die Tokenization erfolgt bei Mistral mittels Byte Pair Encoding (BPE) mit einem Vokabular von 32.000 Tokens Jiang u. a., „Mistral 7B“, wobei LeoLM einen an die deutsche Morphologie angepassten Tokenizer verwendet HessianAI, *LeoLM: Linguistically Enhanced Open Language Model for German*, der besser mit Komposita, Umlauten und anderen sprachspezifischen Besonderheiten umgehen kann.

Transfer Learning 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 Pan und Yang, „A Survey on Transfer Learning“. 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 Radford, Narasimhan u. a., „Improving Language Understanding by Generative Pre-Training“, wodurch das Modell grundlegendes Sprachverständnis, syntaktische Strukturen, semantische Zusammenhänge und implizites Weltwissen erwirbt. In der zweiten Phase wird das Modell mittels Fine-Tuning auf eine spezifische Aufgabe angepasst Howard und Ruder, „Universal Language Model Fine-tuning for Text Classification“, wobei supervised Learning mit aufgabenspezifischen Daten zum Einsatz kommt. Im Kontext dieser Arbeit bedeutet dies die Spezialisierung auf die Transformation von LVB-Verkehrsanweisungen.

Die Vorteile von Transfer Learning sind vielfältig: Howard und Ruder zeigten mit ULMFiT, dass durch Transfer Learning mit nur 100 gelabelten Beispielen eine vergleichbare Performance erreicht werden kann wie mit 10.000 Beispielen

beim Training from-scratch Howard und Ruder, „Universal Language Model Fine-tuning for Text Classification“. Vortrainierte Gewichte dienen als optimaler Startpunkt und beschleunigen das Training erheblich. Zudem führt das bereits vorhandene Sprachverständnis zu besserer Generalisierung auf neue Daten Pan und Yang, „A Survey on Transfer Learning“.

Dennoch bestehen Herausforderungen: Catastrophic Forgetting bezeichnet den Verlust vortrainierter Fähigkeiten während des Fine-Tunings, dem in Kapitel 2.2.4 weiter nachgegangen wird. Der Domain Shift zwischen Pretraining-Daten und Zieldomäne Pan und Yang, „A Survey on Transfer Learning“ kann zu Leistungseinbußen führen, insbesondere wenn sich Vokabular oder Sprachstil deutlich unterscheiden. Bei kleinen Datensätzen besteht zudem die Gefahr des Overfittings, was ebenfalls in Kapitel 2.2.4 behandelt wird.

Für die vorliegende Arbeit ist besonders relevant, dass LeoLM bereits auf umfangreichen deutschen Textkorpora vortrainiert wurde HessianAI, *LeoLM: Linguistically Enhanced Open Language Model for German*, 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 domänenpezifische Anpassung an die fachsprachlichen Anforderungen des Verkehrssektors.

- 2.1.2 NLP-Anwendungen im Verkehrssektor**
 - 2.1.3 Domänen spezifische Sprachmodelle**
 - 2.2 Sprachmodelle und Fine-Tuning**
 - 2.2.1 Fine-Tuning-Methoden**
 - 2.2.2 Prompt Engineering**
 - 2.2.3 Datensätze für Fine-Tuning**
 - 2.2.4 Herausforderungen beim Fine-Tuning**
 - 2.3 Ressourceneffizienz und Modelloptimierung**
 - 2.3.1 Problematik großer Sprachmodelle**
 - 2.3.2 Quantisierung**
 - 2.3.3 Weitere Optimierungsansätze**
 - 2.4 Qualitätssicherung bei KI-generierten Texten**
 - 2.4.1 Evaluationsmetriken für NLP**
 - 2.4.2 Halluzination Detection und Validierung**

3 Anwendungskontext und Anforderungen

3.1 Leipziger Verkehrsbetriebe

3.2 Analyse der Verkehrsanweisungen

3.3 Anforderungen an das System

3.3.1 Funktionale Anforderungen

3.3.2 Nicht-funktionale Anforderungen

4 Modellauswahl: leo-mistral-hessianai-7b

4.1 Anforderungen an das Modell

4.2 Modellvergleich und Auswahlprozess

4.3 Modellvarianten für Deployment

5 Datensatzerstellung und -aufbereitung

5.1 Datenerhebung

5.2 Datensatzentwicklung für Fine-Tuning

5.2.1 Annotationsstrategie

5.2.2 Datenaugmentierung

5.2.3 Datensatzstruktur

5.2.4 Herausforderungen bei kleinem Datensatz

6 Implementierung

6.1 Technische Umsetzung

6.2 Fine-Tuning des Modells

6.2.1 LoRA-basiertes Fine-Tuning

6.2.2 Prompt Engineering und Automatisierung

6.2.3 Quantisierung

6.3 Qualitätssicherung und Validierung

7 Evaluation

7.1 Evaluationsmethodik

7.2 Qualitätsevaluation

7.2.1 Vollmodell vs. Quantisierte Versionen

7.2.2 Automatisierung und Konsistenz

7.3 Gesamtbewertung

8 Diskussion

8.1 Interpretation der Ergebnisse

8.2 Kritische Bewertung

8.2.1 Limitationen der Arbeit

8.2.2 Validität der Ergebnisse

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung der Arbeit

9.2 Beantwortung der Forschungsfragen

9.3 Wissenschaftlicher und praktischer Beitrag

9.4 Ausblick und zukünftige Forschung

9.4.1 Praktische Implikationen

9.4.2 Erweiterungsmöglichkeiten

9.4.3 Offene Forschungsfragen

Abbildungsverzeichnis

1	Problemerkennung	25
2	Performance Kriterien	26
3	Arten von Medien	28
A.1	Hochschullogo	36

Listingverzeichnis

1	Loesungsimplementation Codebeispiel Java	30
2	Anzeigemoeglichkeit Experimente via HTML	32
3	SQL Befehl, zur Extraktion der Ergebnisse	33
A.1	Java Hello World Beispiel	35

Literatur

- Devlin, Jacob u. a. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *arXiv preprint arXiv:1810.04805* (2018).
- HessianAI. *LeoLM: Linguistically Enhanced Open Language Model for German*. <https://huggingface.co/LeoLM>. Accessed: 2025-11-12. 2023.
- Howard, Jeremy und Sebastian Ruder. „Universal Language Model Fine-tuning for Text Classification“. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, S. 328–339.
- Jiang, Albert Q. u. a. „Mistral 7B“. In: *arXiv preprint arXiv:2310.06825* (2023).
- Ouyang, Long u. a. „Training Language Models to Follow Instructions with Human Feedback“. In: *arXiv preprint arXiv:2203.02155* (2022).
- Pan, Sinno Jialin und Qiang Yang. „A Survey on Transfer Learning“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), S. 1345–1359.
- Radford, Alec, Karthik Narasimhan u. a. „Improving Language Understanding by Generative Pre-Training“. In: (2018).
- Radford, Alec, Jeffrey Wu u. a. „Language Models are Unsupervised Multitask Learners“. In: (2019).
- Raffel, Colin u. a. „Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer“. In: *Journal of Machine Learning Research* 21.140 (2020), S. 1–67.
- Vaswani, Ashish u. a. „Attention Is All You Need“. In: *Advances in Neural Information Processing Systems* 30 (2017).
- Wei, Jason u. a. „Finetuned Language Models Are Zero-Shot Learners“. In: *arXiv preprint arXiv:2109.01652* (2021).

Hinweise zur Abgabe

In dieser mit Latex erstellten PDF sind folgende Inhalte enthalten:

- Ein Deckblatt mit Titel, Datum, Prüfer, Matrikelnummer
- Ein Inhaltsverzeichnis
- Wenigstens drei:
 - Kapitel
 - Unterkapitel
 - Unterunterkapitel
- Wenigstens drei Paragraphen
- Wenigstens einem sub-Paragraphen [Siehe Was ist ein Problem?]
- Ein Abbildungsverzeichnis
 - Mit wenigstens drei Bildern
 - * Mit jeweils einem Untertitel
 - * Mit einem abweichenden kurzen Titel für das Abbildungsverzeichnis
 - * Mit jeweils einer Referenzierung der Abbildung im Fließtext des Kapitels
- Ein Listingsverzeichnis (nach Abbildungsverzeichnis)
 - Mit wenigstens zwei Listings
 - * Mit Untertitel
 - * Mit Abkürzung des UT für das Verzeichnis
 - * Mit jeweils einer Referenz im Text darauf.
- Einen Anhang (nach Inhalt und Litverzeichnis) [Siehe Anhang]
 - Mit wenigstens einer Abbildung
 - Mit wenigstens einem Listing
- Zwei wechselnde Hintergrundbilder (jede zweite Seite)
- Ein Literaturverzeichnis vor Anhang und nach Inhalt

- Hervorgehobener Text [Siehe Analyse]
 - Kursiv
 - Fett
 - Unterstrichen
 - In Anführungszeichen
- Listen
 - Numeriert [Siehe Analyse]
 - Nicht numeriert [Siehe Hinweise zur Abgabe]
- Zitate [Siehe Was ist ein Problem?]
 - Indirekt
 - Direkt
 - Zitat aus Internet mit Link der Webseite im Literaturverzeichnis
- Eine Eidesstattliche Erklärung mit Unterschrift als eingefügte PDF am Ende

10 Was ist ein Problem?

Diese Frage selbst stellt schon ein Problem dar. Konkreter sollten wir uns also Fragen: "Wie sieht ein Problem aus, welches ich mit der Ingenieurmethodik lösen kann?"

Einfach ausgedrückt: Ein System mit ungewissen Attributen befindet sich in einem Zustand A und soll mit gegeben Ressourcen in einen besseren Zustand B versetzt werden.

Um diese recht einfache Definition im Kern zu verstehen müssen jedoch einige Elemente in einen dringend notwendigen Kontext gestellt werden. So müssen wir, um eine Problematik mit der Ingenieurmethodik auf zu arbeiten folgende Begriffe definieren: Veränderung, Ressourcen, am Besten & Ungewiss.

10.1 Veränderung

Veränderung ist der Übergang von einer Situation A in eine Situation B wobei sich Situation A von B unterscheidet. In unserem Falle sind Situation A und B allerdings nicht fest definiert sondern müssen erst erkannt werden. Herauszufinden wie Situation B aussieht und wie der Übergang herbei zu führen ist ist Aufgabe des Ingenieurs. Hierzu muss er allerdings Situation A möglichst genau definieren, eine Aufgabe die ihm selten vom Auftraggeber abgenommen wird, auch wenn er dies eigentlich tun sollte. So muss der Ingenieur erstmal das genaue Problem feststellen um dann für eben dieses Problem nicht die Lösung sondern eine Lösung zu finden. Eben diese Lösung sollte dann die passendste aller möglichen Lösungen sein.

10.1.1 Am Besten

Der Begriff "am Besten" ist im Falle des Ingenieurs nicht die ideale Lösung des Problemes in einem ansonsten leeren Raum sondern ein möglichst optimaler Kompromiss aus allen Faktoren und Kriterien. Da eben diese Kriterien von Person zu Person zumeist unterschiedlich stark gewichtet werden ist es die Aufgabe des Ingenieurs die allgemeine Gewichtung eben dieser Kriterien durch die Gesellschaft zu ermitteln. Da auch dies ein subjektiver Prozess ist (bei dieser Evaluation spielen die persönlichen Kriterien des Ingenieurs natürlich auch eine Rolle) kann es keine ideale Lösung geben. Die "beste" Lösung die im Laufe dieses Prozesses zu finden ist ist die, die möglichst viele Kriterien einfließen lässt und abhängig der Gewichtung möglichst viele Teilnehmer zufriedenstellt.¹

1. Vgl. bock2001

10.1.2 Ungewissheit

Ein großer Teil, nein, gar der Kern eines jeden Ingenieurs-Projektes ist es mit unvollständigen Informationen auf ein schwammig definiertes Ziel mit unbekannt vielen Faktoren und Möglichkeiten zu zu arbeiten. Wäre dies anders, also die Informationen vollständig, die Aufgabe klar definiert und alle Faktoren bekannt wäre das Ganze nicht mehr als eine wissenschaftliche Formel zu lösen. Alle Variablen werden Eingesetzt und am Ende steht das Ergebnis. Sollte sich im Laufe dessen rausstellen dass sich Fehler eingeschlichen haben kann man diese korrigieren und die Variablen nochmals einsetzen. Mögen sich eben diese Variablen auch ändern, die Formel und Prozedur bleiben gleich. Eben diese Sicherheit ist bei einem Ingenieur-Problem jedoch nicht gegeben, da bis zum Schluss kein vollständiges Bild der Ausgangs- und Endlage bekannt ist. Die Aufgabe des Ingenieurs ist es also diese Ungewissheit der Situation zu minimieren, in dem er möglichst viele Faktoren und Kriterien in seine Überlegungen mit einfließen lässt und so einen optimalen Kompromiss findet.²

10.2 Der Kern des Ingenieursproblems

Zusammengefasst sieht jedes Problem der Ingenieurmethodik wie folgt aus: Ein System mit ungewissen Attributen befindet sich in einem Zustand A und soll mit gegebenen Ressourcen in einen besseren Zustand B versetzt werden.³

2. koen2013
3. method

11 Analyse

Die Aufgabe der Analysephase ist es, eingehendes Verständnis über alle Komponenten des Problemgebietes zu erlangen, sodass ein einzelnes, spezifisches und realistisches Ziel formuliert werden kann.

Das oberste Ziel der Analyse sollte immer sein, die vorerst viel umfassende und grobe Zielsetzung zu einer einzigen, spezifischen Aufgabe zu reduzieren. Die zuerst weite Problembeschreibung wird durch die Aufstellung leitender Thesen systematisch eingeschränkt, bis ein enges, aber hoch detailliertes Ziel im finalen Schritt der Analyse formuliert werden kann.

11.1 Beschreibung des Problems

Projekte können in zwei Untergruppen eingeteilt werden. Einerseits gibt es das Forschungsprojekt, dessen Ziel die Gewinnung neuen Wissens ist. Daneben gibt es noch das Entwicklungsprojekt, bei dem man bereits existierendes Wissen nutzt, um ein neues Gerät zu entwickeln oder einen bestimmten Effekt zu erzeugen. Häufig kann man ein Projekt allerdings nicht strikt zu einem der beiden Gruppen zuordnen. Oftmals führt die Entwicklung eines neuen Geräts im Laufe seiner Entwicklungsphase auch zu neuen Erkenntnissen. Manchmal ist es auch nötig ein neues Gerät zu entwickeln, damit neues Wissen überhaupt zugänglich wird. Zweiteres trifft allerdings nur eher selten ein. Die Beschreibung eines Problems wird in der folgenden Grafik verdeutlicht.



Abbildung 1: Der Moment des Erkennens eines Problems

Sollte es schwer fallen, das Problem mit einer kurzen, präzisen Frage oder Aussage zu umschreiben, enthält das Projekt möglicherweise mehr als ein Problem. Dies ist ein häufig auftretender Grund für Verwirrung im Projektplan, denn unterschiedliche Probleme benötigen fast immer auch unterschiedliche Lösungen. Die dadurch verworrenen und sich teilweise widersprechenden Projektziele können zu großen Komplikationen führen und verheerende Auswirkungen auf den Erfolg des Projekts haben, sollten sie sich auch durch die Restlichen Projektphasen ziehen. Daher sollte am Anfang der Analyse eine möglichst detaillierte Zusammenfassung des Problems angefertigt werden. Darin beinhaltet sind auch ausreichende Hintergrundinformationen und eine Motivation zur Behebung des Problems, um das Projekt im richtigen Kontext einordnen zu können.

11.2 Performance Kriterien festlegen

„Performance Kriterien sind Bedingungen, die jede vorgeschlagene Lösung zu einem Problem erfüllen muss.“ Ziel ist hier, den in der Abbildung 1 gezeigten Moment möglichst zu vermeiden. Dabei sollte darauf geachtet werden, dass die Kriterien weder zu streng, noch zu offen sind. Das Beispiel sei hier das Festlegen einer Deadline für einen Teil eines Projekts. Die vorgeschlagene Lösung hier wäre drei Monate. Sind die Kriterien zu eng, würde das Projekt nach Ablauf der festgelegten Zeit gestrichen werden, auch wenn die eigentliche Dauer gerade mal drei Monate und ein Tag gewesen wären. Werden die Kriterien jedoch zu aussagelos gesetzt, könnte das Teilprojekt potenziell nie ein Ende finden und das gesamte Projekt gerät in nicht aufholbaren Verzug. Beide Ausgänge sind zu vermeiden. Die meisten Entwicklungsprojekte sind jedoch im Voraus schon von den festen, strengen Bedingungen und Parametern der Anwendungsdomäne eingegrenzt. Das folgende Diagramm 2 liefert einen Überblick über die wichtigsten generell anerkannten Kriterien.

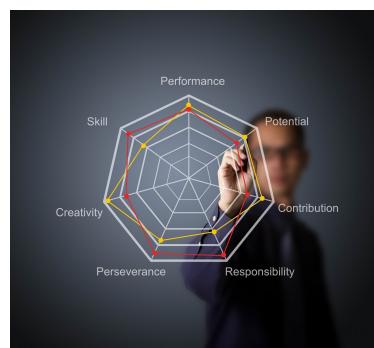


Abbildung 2: Die wichtigsten Performance Kriterien im Bereich des R&D Sektors

11.3 Themenverwandte Arbeiten untersuchen

Ist das Problem dann formal und präzise beschrieben und alle grundsätzlichen Performance Kriterien festgelegt, so kann das Projektteam nun damit anfangen, möglichst viele Informationen über vorhergehende Projekte in einem ähnlichen oder im gleichen Rahmen zu sammeln. Der wohl bedeutendste Grund dafür ist es zu verhindern, "Das Rad neu zu erfinden." Sollten schon Lösungen für Teile, oder sogar das Projekt im Ganzen bestehen, ist es fast immer einfacher und billiger die Lösung oder das Gerät zu kaufen, als der Versuch es zu replizieren. Es gibt verschiedene Arten, um nach relevanten Quellen zu suchen. Einige davon sind hier zusammengefasst:

1. Professionelle Tagebücher
2. Konferenzdokumentationen
3. Bücher und Monographien
4. Professionelle Studien
5. Zeitungsartikel
6. Diskussionen mit Kollegen
7. Reverse engineering
8. Digitale Medien
 - (a) TV Berichte
 - (b) Radiobeiträge
 - (c) Ressourcen aus dem Internet
 - i. Webseiten
 - ii. Datenbanken
 - iii. Videos

Die Liste wird in Der Abbildung 3 um weitere Punkte ergänzt. Unabhängig von der Quelle muss die Validität immer gründlich überprüft werden. Zweitquellen, wie z.B. Zeitungsartikel, müssen immer auf ihre ursprüngliche Quelle zurückführbar sein, um die Echtheit der Information versichern zu können. Die Quellen, die als Vorwissen in ein Projekt gebracht werden, müssen klar und strukturiert zitiert werden.



Abbildung 3: Ein Pfahl, der genau das gleiche wie der obige Text aussagt, nur cooler

11.4 Ziel formulieren

Dies ist der letzte Schritt in der Analysephase. Nachdem das Problem klar definiert, die Performance Kriterien festgelegt und verwandtes Material gründlich untersucht wurde, ist die Aufgabendomäne nun angemessen fokussiert und eingegrenzt. Jetzt kann ein spezifisches und klares Ziel für das Projekt bestimmt werden. Das Ziel ist ein Ausdruck von dem, was das Projekt im Bestfall erreichen soll. Bevor zur nächsten Phase übergegangen wird, müssen noch zwei Faktoren beachtet werden. Als erstes, das Ziel ist die Basis, an der Erfolg oder Versagen des Projekts gemessen werden. Auch die Leistung des Projektteams wird daran gemessen, wie gut es das Ziel erreicht hat. Als zweites, auch wenn andere Komponenten des Projekts relativ unkompliziert angepasst werden können, benötigt es für eine Änderung des Ziels explizite Erlaubnis der Management Ebene, was zur Verschwendungen großer Zeit- und Geldressourcen führen kann und deshalb vermieden werden sollte. Umso wichtiger ist es, schon von Anfang an ein allgemein anerkanntes Ziel festzulegen.

12 Synthese

Das Ergebnis der Synthesephase sollte die Anwendung der designten Lösung sein, um die gesetzten Ziele zu erreichen und die die Aufgabe begleitende Hypothese zu validieren.

Auch dieser Prozess kann in 4 kleinere Funktionen unterteilt werden aber anders als in der Analysephase, muss die Reihenfolge aller Schritte genau eingehalten werden. In dieser Phase werden alle vorher aufgestellten Thesen, Faktoren, Bedingungen, etc. mit Hilfe von Experimenten in die Praxis umgesetzt, um erste konkrete Ergebnisse zu erzielen.

12.1 Lösungsansätze implementieren

Es gibt zwei Arten, auf die eine Lösung implementiert werden kann, entwickeln und kaufen. Ein Beispiel für eine selbst entwickelte Lösung kann im Listing 1 betrachtet werden. Sollte es den Anforderungen entsprechen, ist es fast immer effektiver die Lösung zu erwerben, als sie selbst zu entwickeln. Dies trifft auch zu, sollte die erworbene Komponente nur zu einem Teil zur Lösung beitragen. Besteht ein Teil der Lösung aus einer Datenbank, sollte der Aufwand betrieben werden, um die Daten vollständig, roh und unbearbeitet zu erhalten. Dies trifft vor allem zu, wenn die Daten von einer anderen Forschungsgruppe oder Firma kommen. Häufig treten im Zusammenhang dessen auch politische oder Eigentums Probleme auf. Von Drittparteien erhaltene Daten sollten generell immer mit einer gewissen Skepsis behandelt werden.

```
5
6     package com.tutego.insel.ds.observer;
7
8     import java.util.*;
9
10    public class Party
11    {
12        public static void main( String[] args )
13        {
14            Observer achim      = new JokeListener( "Achim" );
15            Observer michael   = new JokeListener( "Michael" );
16            JokeTeller chris   = new JokeTeller();
17
18            chris.addObserver( achim );
19
20            chris.tellJoke();
21            chris.tellJoke();
22
23            chris.addObserver( michael );
24
25            chris.tellJoke();
26
27            chris.deleteObserver( achim );
28
29            chris.tellJoke();
30        }
31    }
32
```

Listing 1: Beispiel einer Loesungsimplementation nach den festgelegten Vorschriften

12.1.1 Experimente Designen

Der Sinn hinter diesem Schritt ist es, eine Reihe von Experimenten zu designen, dessen Resultate benutzt werden, um zu messen wie gut eine Aufgabe erfüllt wurde. Ein Experiment erfasst Daten um den Erfolg einer konzipierten Lösung unter kontrollierbaren Bedingungen im Labor zu testen.

Vor dem Beginn der Experimente sollten alle aufgestellten und gesammelten Faktoren, Regeln, Bedingungen und Beobachtungen auf Vollständigkeit und Korrektheit überprüft werden, da jede dieser Komponenten einen fundamentalen Einfluss auf die Planung und Durchführung der Experimente haben kann. Wurde diese Liste noch ein letztes mal verifiziert, kann nun mit der Planungsphase für die Experimente fortgefahren werden. Zuerst sollte sichergestellt werden, dass das Labor, in dem das Experiment stattfinden wird, frei von nicht-essentiellen Objekten ist. Der negative Einfluss von emotionalen Reaktionen der Testsubjekte sowie der Prüfer kann oft nicht vollkommen verhindert werden. Deshalb sollte man darauf achten, dass jedes Element im Labor unabdingbar für das Experiment ist. Einmal richtig angeordnet, sollte danach großer Wert darauf gelegt werden, dass niemand außer den am Experiment beteiligten Personen das Labor betritt. Außenstehende können ohne ihr Wissen kleinste Veränderungen am Arbeitsraum durchführen, z.B. ein Gerät um wenige Millimeter verschieben, und damit die Ergebnisse des Experiments vollkommen aussagelos machen. Das schlimmste daran ist, dass ein solcher Fehler, wenn überhaupt, nur sehr schwer zurückverfolgt werden kann, wodurch das ganze Projekt in Gefährdung stehen könnte. Deshalb sollte das Labor in Abwesenheit des Personals immer verriegelt und klar und deutlich für Außenstehende gekennzeichnet sein, um die Chance auf derartige Fehler zu minimieren. Das eigentliche Experiment besteht aus mehreren Elementen. Angefangen wird bei dem Personal, dass das Experiment durchführen wird. Diese haben einen genauen Überblick über alle bisher gesammelten Daten sowie die nötige Expertise um die Experimente auch durchzuführen. Lebendige Teilnehmer eines Experiments werden Kohorte genannt, ihr Gegenpol sind die Stichproben. Die Durchführung eines Versuchs mit einer schrittweisen Veränderung der Faktoren, bis alle möglichen Faktorkombinationen abgedeckt sind, heißt Block Design. Um die vorgesehene Funktion aller Faktoren bestätigen zu können, werden sogen. Control Trials durchgeführt. Dabei wird die Performance von einem Set von Faktoren in Abwesenheit eines anderen Sets von Faktoren gemessen, um die isolierten Effekte des Sets festhalten zu können.

Die aus Experimenten erhaltenen Daten sollten auf mehr als einem Medium gespeichert sein, um den Verlust im Falle eines Ausfalls eines der Medien zu verhindern. Datenschutz sollte eine der obersten Prioritäten sein. Am besten sollten fertig bearbeitete Daten irgendwo im Internet gespeichert werden, damit sie für jeden zugänglich sind. Dies könnte mit Hilfe einer Webseite, wie im Listing 2 passieren. Es empfiehlt sich, die Daten zu verschlüsseln und diesen Schlüssel nur an die Personen weiterzugeben, die unbedingt Zugriff auf die Daten benötigen. Werden über die schon bestehenden Speicherplätze der Daten hinaus weitere Kopien angefertigt, so muss dies unbedingt Dokumentiert werden, um unkontrollierte Verbreitung der Daten zu verhindern. Sind alle Daten einmal gespeichert, sollten diese umgehend mit Hilfe des Copyright weiter geschützt werden. In der Regel ist es sinnvoll, zu viele Daten über das Experiment gespeichert zu haben als zu wenig.

```
5      <!DOCTYPE html>
6      <html>
7          <body>
8
9
10         <h1>My First Heading</h1>
11         <p>My first paragraph.</p>
12
13     </body>
14 </html>
15
```

Listing 2: Die Grundlage einer HTML Seite, zum Teilen der designten Experimente

12.1.2 Experimente durchführen

Jetzt gilt es nur noch, die bereits fertig geplanten Experimente durchzuführen. Dies sollte genauestens nach Plan passieren, damit Fehler oder unerwartete Ergebnisse in den Plänen gefunden und behoben werden können. Mislungene Experimente sollten als Hilfen gesehen werden, um die Pläne zu überarbeiten und dem Erfolg einen Schritt näher zu kommen. Um eine fehlerfreie

12.1.3 Ergebnisse extrahieren

Oft können die direkten Ergebnisse eines Experiments nicht genutzt werden, um Schlussfolgerungen zu ziehen. Zuerst müssen die Resultate reduziert werden, also umgewandelt und kombiniert, damit die entstehenden Werte in Form der gewählten Performance Metrik dargestellt werden können. Wurden bis jetzt alle Richtlinien beachtet, sollten die Daten über einen simplen SQL Befehl, gezeigt im Beispiel 3 ohne weitere Probleme extrahiert werden können. Im Grunde können nur zwei Arten von Werten direkt gemessen werden: Distanzen und Zählungen. Weil die reduzierten Daten oftmals modifiziert oder korrigiert werden müssen, sollten immer die rohen und die reduzierten Ergebnisse des Experiments dokumentiert werden.

```
5
6      USE AdventureWorks2012;
7      GO
8      SELECT *
9      FROM Production.Product
10     ORDER BY Name ASC;
11     -- Alternate way.
12     USE AdventureWorks2012;
13     GO
14     SELECT p.*
15     FROM Production.Product AS p
16     ORDER BY Name ASC;
17     GO
18
```

Listing 3: SQL Befehl, um Ergebnisse aus dem Experiment zu extrahieren

Anhang

A Nachbereitung

Als allererstes, hier noch die letzte Referenz zum folgenden Listing im Anhang:
A.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. Überflüssigestellen 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 A.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.

Loreum 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 Loreum ipsum dolor sit amet. Loreum 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 Loreum ipsum dolor sit amet. Loreum 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 A.1: Das Logo unserer Hochschule

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig 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