

Bachelorarbeit

im Studiengang Informatik

Thema: Konzeption und prototypische Umsetzung eines KI-gestützten Arbeitsablaufs zur Vorgangserfassung in @rtus

eingereicht von: Noah Zepner

eingereicht am: 06.10.2025

Erstprüfer: Prof. Dr. Robert Manzke

Zweitprüfer: Dr.-Ing. Martin Toepfer

Abstract

Inhaltsverzeichnis

Abstract	ii
Abkürzungsverzeichnis	v
I. Einleitung	1
A. Motivation	1
B. Ziel	1
C. Hypothese	1
D. Aufbau der Arbeit	2
II. Projektkontext und Grundlagen	3
A. Bisherige Forschungsarbeiten	3
B. Überblick über das VBS @rtus	4
C. Ergebnisse der vorangegangenen Machbarkeitsstudie	4
D. Grundlagen zu LLMs	4
1) Funktionsweise	4
2) Fehlerquellen und Limitationen	6
E. Kontextbasierte Generierung	7
F. Datenformate	9
III. Analyse und Anforderungsdefinition	10
A. Anforderungen der Polizei an einen KI-basierten Arbeitsablauf zur Vorgangserfassung	10
B. Funktionale Anforderungen	10
C. Nicht-funktionale Anforderungen	10
IV. Konzeption	11
A. Architekturentwurf	11
B. Datenmodellierung	11
C. Einbindung des LLMs in @rtus	11
D. Festlegung der Evaluationskriterien	11
V. Implementierung	12
A. Technologiewahl	12
B. Integration des LLMs in das @rtus Backend	12
C. Anbindung an vorhandene Datenstrukturen	12
VI. Evaluation und Tests	13
A. Testumgebung und Methodik	13
B. User-Studie: UI/UX	13
C. Ergebnisse	13
VII. Diskussion	14
A. Bewertung der Ergebnisse anhand der Hypothese	14
B. Grenzen des Systems	14

Inhaltsverzeichnis

C. Praktische Relevanz für den polizeilichen Arbeitsalltag	14
VIII. Fazit und Ausblick	15
A. Zusammenfassung der Ergebnisse	15
B. Beantwortung der Forschungsfragen	15
C. Weiterführende Ansätze und mögliche Weiterentwicklungen	15
Literaturverzeichnis	I
Anhang	IV

Abkürzungsverzeichnis

GPT	Generative Pre-Trained Trans- former	NLP	Natural Language Processing
HAW Kiel	Hochschule für Angewandte Wissenschaften	NLU	Natural Language Understan- ding
KI	Künstliche Intelligenz	RAG	Retrival-Augmented Generati- on
LLM	Large Language Model	VBS	Vorgangsbearbeitungssystem

I. Einleitung

A. Motivation

Das Vorgangsbearbeitungssystem(VBS) @rtus ist das zentrale System zur Erfassung, Strukturierung und Bearbeitung polizeilicher Vorgänge in Schleswig-Holstein sowie der Bundespolizei und zunehmend mehr Landespolizeien. Bisher ist in @rtus kein KI-gestützter Arbeitsprozess integriert. Außerhalb von @rtus werden KI-Komponenten jedoch zunehmend in Anwendungen mit ähnlichen Arbeitsabläufen eingesetzt, beispielsweise zur automatischen Extraktion strukturierter Informationen aus Freitexten, Formularen oder Medien.

Im polizeilichen Kontext entstehen große Mengen unstrukturierter Daten, nicht nur im Rahmen von Ermittlungen, sondern auch in andere Kanälen wie der *Online-wache* und @rtus-Mobile. Die manuelle Überführung in strukturierte Fachobjekte ist zeitaufwendig und fehleranfällig. Durch den Einsatz eines Large Language Models (LLMs) kann dieser Prozess erleichtert werden, indem die anfallenden Daten ausgewertet und dem Sachbearbeiter ein vorausgefüllter Vorgang zur weiteren Bearbeitung vorgelegt wird. Gleichzeitig stellt die Integration in eine bestehende, sicherheitskritische Software, deren Komplexität über viele Jahre der Entwicklung dynamisch gewachsen ist, eine Herausforderung dar.

B. Ziel

Ziel dieser Arbeit ist die prototypische, technische Vorbereitung einer KI-gestützten Verarbeitung unstrukturierter Daten in @rtus durch einen neuen Arbeitsablauf. Die Arbeit grenzt sich damit vom vorhergehenden Studentenprojekt („Ein LLM für @rtus“) ab. Statt die dort erprobte Interaktion direkt zu integrieren, fokussiert sich diese Arbeit auf die technische Umsetzung einer LLM-basierten Extraktions- und Aufbereitungspipeline, die unstrukturierte Eingaben in ein für @rtus auswertbares Format überführt. Der Prototyp soll als Wegbereiter dienen, um eine spätere produktive Integration zu ermöglichen.

C. Hypothese

Die prototypische Integration eines LLMs in das VBS @rtus ermöglicht die automatisierte Überführung unstrukturierter Eingaben aus Quellen wie der *Onlinewache*

C. Hypothese

und @rtus-Mobile in strukturierte, systemkompatible Fachobjekte für nachgelagerte Verarbeitungsschritte.

D. Aufbau der Arbeit

Die Arbeit beginnt mit einem kurzen Überblick über bereits vorhandene Forschungsarbeiten, auf denen sie inhaltlich aufbaut. Anschließend wird der Kontext der Arbeit präzisiert, indem ein Überblick über das VBS @rtus gegeben und die Machbarkeitsstudie aus dem *Projekt Informatik* („Ein LLM für @rtus“) betrachtet wird. Zur Sicherung der Verständlichkeit werden danach grundlegende theoretische Konzepte eingeführt.

In Kapitel III werden die Anforderungen an den zu entwickelnden Arbeitsablauf dargestellt. Zunächst werden die erforderlichen Bedingungen der Polizei an eine KI-basierte Vorgangserfassung erhoben; anschließend werden funktionale und nicht-funktionale Anforderungen bereitgestellt.

In Kapitel IV wird die Konzeption der Integration erläutert. Es werden der Architektorentwurf und die Datenmodellierung vorgestellt sowie die Einbindung eines LLM in die Codebasis von @rtus konzeptionell vorbereitet. Zudem werden Evaluationskriterien festgelegt, um anhand der Integration die Fragestellungen dieser Arbeit beantworten zu können.

Kapitel V beschäftigt sich mit den technischen Details der Implementierung und stellt die aufkommenden Probleme sowie die gefundenen Lösungen dar.

In Kapitel VI werden die Testumgebung und die Methodik beschrieben und darauf aufbauend Tests durchgeführt. Die Ergebnisse werden anschließend anhand der Evaluationskriterien beurteilt.

In Kapitel VII werden die Ergebnisse im Lichte der Hypothese diskutiert; dabei werden die Grenzen des Systems und die praktische Relevanz für den polizeilichen Arbeitsalltag herausgearbeitet.

Kapitel VIII schließt mit einem Fazit ab, beantwortet die Forschungsfragen und gibt einen Ausblick auf mögliche Weiterentwicklungen.

II. Projektkontext und Grundlagen

A. Bisherige Forschungsarbeiten

Zur Einordnung dieser Arbeit in den aktuellen Stand der Forschung wird in diesem Abschnitt eine Literaturübersicht vorgenommen.

Mit der KI-Welle ab 2021/2022 hat sich die Nutzung von LLMs zur Informationsextraktion dynamisch entwickelt. Obwohl LLM-basierte Ansätze hohe Aufmerksamkeit erhalten, handelt es sich in vielen Anwendungsdomänen weiterhin um ein junges Forschungsfeld.

Für den spezifischen Kontext der Extraktion strukturierter Informationen aus polizeilichen Freitexten ergab eine systematische Suche (siehe Literaturrechercheprotokoll, Seite IV) nur wenige peer-reviewte Arbeiten. Einige relevante Ausnahmen stellen die Beiträge [1], [2] und [3] dar. Darüber hinaus existieren ältere Studien mit klassischen NLP-Verfahren [4], [5] sowie jüngere, überwiegend als Preprints verfügbare Arbeiten zur LLM-basierten Extraktion in anderen Domänen, z. B. Sport [6], Landwirtschaft [7] und Medizin [8]). Methodisch sind diese Ansätze teilweise übertragbar. Ergänzend liefern [9] und [10] praxisrelevante Hinweise zur technischen Integration von LLMs in Unternehmensumgebungen (z. B. Hosting, Retrieval-Augmented Generation (RAG), Datenvorbereitung), die für die Architektur dieser Arbeit relevant sind.

Auch die Verbesserung der Ergebnisse durch den Ansatz eines mehrstufigen Arbeitsablaufs wurde bereits im medizinischen Fachgebiet getestet. [11]

Angesichts des explorativen Charakters dieser Arbeit dienen die genannten Arbeiten der kontextuellen Einordnung und zur Ableitung von Methoden (z. B. Prompting/RAG, Evaluationsmetriken). Aufgrund abweichender Domänen, Sprachen und Datenlagen werden sie nicht als direkte Baselines für die Evaluation herangezogen, fließen jedoch in die Begründung der Architektur- und Designentscheidungen ein.

B. Überblick über das VBS @rtus

C. Ergebnisse der vorangegangenen Machbarkeitsstudie

D. Grundlagen zu LLMs

1) Funktionsweise

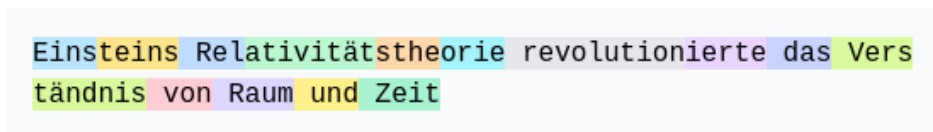
LLMs sind eine Klasse von Sprachmodellen, die auf der von Google vorgestellten Transformer-Architektur[12] aufbauen und als neuronale Netze auf immensen Textdatenmengen trainiert sind.[13] Obwohl LLMs eine große Rolle dabei gespielt haben, generative KI in das Bewusstsein der Öffentlichkeit zu rücken, haben Unternehmen wie IBM bereits in den Jahren vor 2017 daran gearbeitet, Sprachmodelle für Natural Language Understanding (NLU) und Natural Language Processing (NLP) zu verbessern – damals noch nicht auf Basis der Transformer-Architektur. Im Kern stellen LLMs den neuesten Durchbruch in NLP und Künstliche Intelligenz (KI) dar. Sie sind so konzipiert, dass sie Inhalte ähnlich wie Menschen verstehen und generieren können, um eine breite Palette von Anwendungsfällen abzudecken. Sie sind in der Lage, aus dem Kontext zu schließen, kohärente und kontextbezogene Antworten zu geben, in andere Sprachen zu übersetzen, Texte zusammenzufassen, Fragen zu beantworten und bei kreativen Schreib- oder Codegenerierungsaufgaben zu helfen. [13]

Grundsätzlich lässt sich ein LLM in vier wesentliche Funktionen unterteilen: [14]

1. Tokenisierung
2. Einbettung
3. Vorhersage
4. Dekodierung

Die Tokenisierung beschreibt ein Verfahren, bei dem ein Text in viele kleinere Teile zerlegt wird. Dafür gibt es unterschiedliche Techniken, z. B. die Aufteilung nach einzelnen Wörtern. Die Wahl einer Tokenisierungstechnik ist eine Abwägung zweier Aspekte: zum einen die Gesamtzahl der möglichen Token, zum anderen der semantische Informationsgehalt. Aktuelle LLMs setzen auf Tokenisierungsverfahren, die Subwörter (Teilwörter) produzieren.

1) Funktionsweise



Einsteins Relativitätstheorie revolutionierte das Verständnis von Raum und Zeit

Abbildung 1: Ein Beispiel für Tokenisierung (Erstellt mit <https://tiktokenizer.vercel.app/>)

Im zweiten Schritt werden Token auf Vektoren abgebildet – die sogenannte Einbettung (engl. Embedding). Dabei werden die Vektoren so berechnet, dass semantisch ähnliche Informationen zu ähnlichen Vektoren führen. Das ermöglicht neuronalen Netzen, mit Texten zu arbeiten. In der Transformer-Architektur spielt zusätzlich zur Semantik auch die Position im Satz eine Rolle.

Nach der Einbettung bestimmt der Transformer, welche Teile des Kontexts für ein bestimmtes Token wichtig sind.[15] Dieses Verfahren heißt Selbstaufmerksamkeit (Self-Attention): Für jedes Token wird gewichtet, wie stark es die übrigen Token im selben Kontextfenster berücksichtigen sollte. Anschaulich gefragt jedes Token gewissermaßen: „Welche anderen Wörter sind für meine Bedeutung gerade relevant?“ Ein Beispiel: In dem Satz „Das Tier überquerte die Straße nicht, weil es zu müde war.“ kann das Pronomen „es“ mehrdeutig sein. Die Selbstaufmerksamkeit hilft dem Modell, die passenden Bezüge zu finden – hier typischerweise zum „Tier“ und nicht zur „Straße“ –, indem relevante Wörter höher gewichtet werden. Technisch geschieht dies über sogenannte Queries, Keys und Values, oft in mehreren parallelen „Köpfen“ (Multi-Head Attention). Positionsinformationen werden separat kodiert (z. B. mittels Positional Encodings oder gelernter Positions-Einbettungen), damit das Modell auch Wortreihenfolge berücksichtigen kann.[15]

Der dritte Schritt ist die Vorhersage des nächsten Tokens. Viele moderne LLMs werden autoregressiv trainiert (z. B. GPT-Modelle) und lernen, das nächste Token im Kontext vorherzusagen. Es gibt aber auch andere Trainingsziele (Masked LM, Seq2Seq). Die Modellgröße (Anzahl der Parameter) wird erhöht, um mehr Muster aus großen Datenmengen zu erfassen und bessere Generalisierung zu erreichen. [14] Genau hier setzen die meisten modernen LLMs auf neuronale Netze mit der Transformer-Architektur (z. B. das *T* in *Generative Pre-Trained Transformer (GPT)*).

Um zu einem möglichst guten Ergebnis zu kommen, wird im letzten Schritt auf Basis der berechneten Wahrscheinlichkeiten das nächste Token ausgewählt. Bei der Dekodierung gibt es verschiedene Strategien: Greedy Search, Beam Search,

¹Mehr Informationen zu den unterschiedlichen Strategien: <https://huggingface.co/blog/mlabonne/decoding-strategies>

1) Funktionsweise

Top-k, Top-p (Nucleus) Sampling und Temperatursteuerung¹ — jede Methode hat Vor- und Nachteile hinsichtlich Vielfalt, Kohärenz und Berechenbarkeit.[16]

2) Fehlerquellen und Limitationen

Trotz ihrer hohen Leistungsfähigkeit stoßen LLMs in mehreren Bereichen an klare Grenzen. Das Training erfordert enorme Rechenressourcen, spezialisierte Hardware und viel Energie. Mit wachsender Modellgröße steigt die technische Komplexität, etwa bei Stabilität, Speicherbedarf oder der Verteilung über viele Recheneinheiten, während der Leistungszuwachs nach dem Prinzip abnehmender Erträge immer geringer ausfällt. [17] Diese Anforderungen machen den gesamten Entwicklungs- und Einsatzprozess kostenintensiv und schränken die Zugänglichkeit großer Modelle erheblich ein. [17]

Da LLMs auf großen Mengen menschlicher Sprache beruhen, übernehmen sie leicht vorhandene Verzerrungen und stereotype Darstellungen. Dadurch können sie unangemessene oder fehlerhafte Inhalte erzeugen. Maßnahmen wie sorgfältige Datenaufbereitung, die Ausrichtung an menschlichen Präferenzen (etwa mit RLHF) und laufende Sicherheitsbewertungen sollen solche Effekte verringern. [15], [17] Eine kontinuierliche Prüfung und Nachjustierung im Betrieb ist dabei entscheidend, um faire und verlässliche Ergebnisse sicherzustellen. [15], [17]

Auch Datenschutz und Nachvollziehbarkeit bleiben kritisch. Große Modelle sind anfällig dafür, Trainingsinhalte unbeabsichtigt wiederzugeben, und ihre internen Entscheidungsprozesse sind kaum transparent. Eine verantwortungsvolle Nutzung setzt daher erklärbare Verfahren, Datenfilterung und kontinuierliche Audits voraus. [17] Nur durch nachvollziehbare und überprüfbare Mechanismen kann langfristig Vertrauen in diese Systeme entstehen. [17]

Fehlerhafte oder erfundene Antworten, sogenannte Halluzinationen, gelten zudem als besonders problematisch. Sie entstehen häufig durch begrenzte Kontextlängen oder veraltetes Wissen. Verbesserte Kontextmodelle und erweiterte Zugriffsmöglichkeiten auf aktuelle Informationen bilden hier einen wichtigen Ansatzpunkt für zukünftige Entwicklungen, insbesondere in Verbindung mit Retrieval-Augmentation. [15], [17]

E. Kontextbasierte Generierung

Um LLMs sinnvoll für den Anwendungsfall dieser Arbeit nutzen zu können, müssen zwei Probleme adressiert werden: Erstens muss der Kontext — also die unstrukturierte Eingabe und das gewünschte Ausgabeformat — dem LLM bekannt gemacht werden; zweitens müssen Halluzinationen, insbesondere in sensiblen Bereichen wie dem polizeilichen Kontext, minimiert werden.

Eine verbreitete Lösung ist die kontextbezogene Generierung von Antworten. Hierfür wird häufig die RAG-Architektur eingesetzt. [18], [19]

Die Kernidee von RAG ist, zunächst relevante Inhalte aus einem externen Speicher zu ermitteln und dann auf Basis dieser Inhalte eine Antwort zu generieren. [19], [20] Vereinfacht gliedert sich der Ablauf in wenige, klar getrennte Schritte:

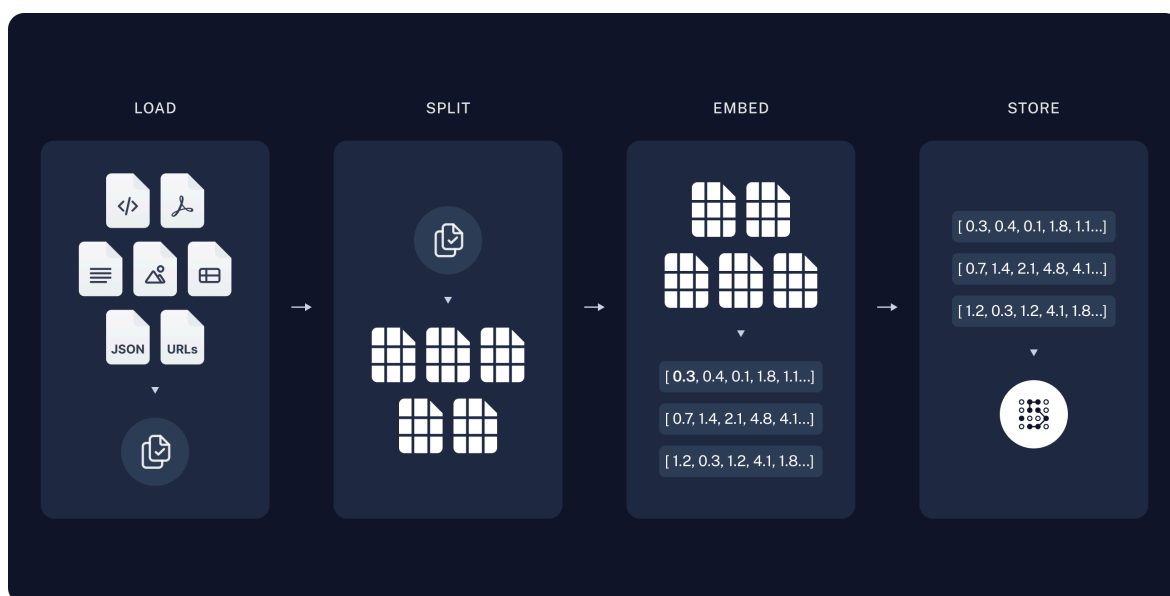


Abbildung 2: Beispielgrafik für das RAG-Indexing aus der LangChain-Dokumentation.

Quelle: <https://github.com/langchain-ai/docs> (MIT License)

Copyright (c) 2025 LangChain. Voller Lizenztext im Anhang, Seite IV

Zunächst wird der Wissensbestand aufbereitet: Längere Dokumente werden in inhaltlich sinnvolle Abschnitte („Chunks“) zerlegt und mit einem Embedding-Modell in dichte Vektoren überführt. Diese Vektoren werden in einem semantischen Index (z. B. Vektordatenbank / ANN) abgelegt. Dadurch lassen sich später nicht nur Schlüsselwörter, sondern die semantische Bedeutung einer Anfrage wiederfinden. [19], [21]

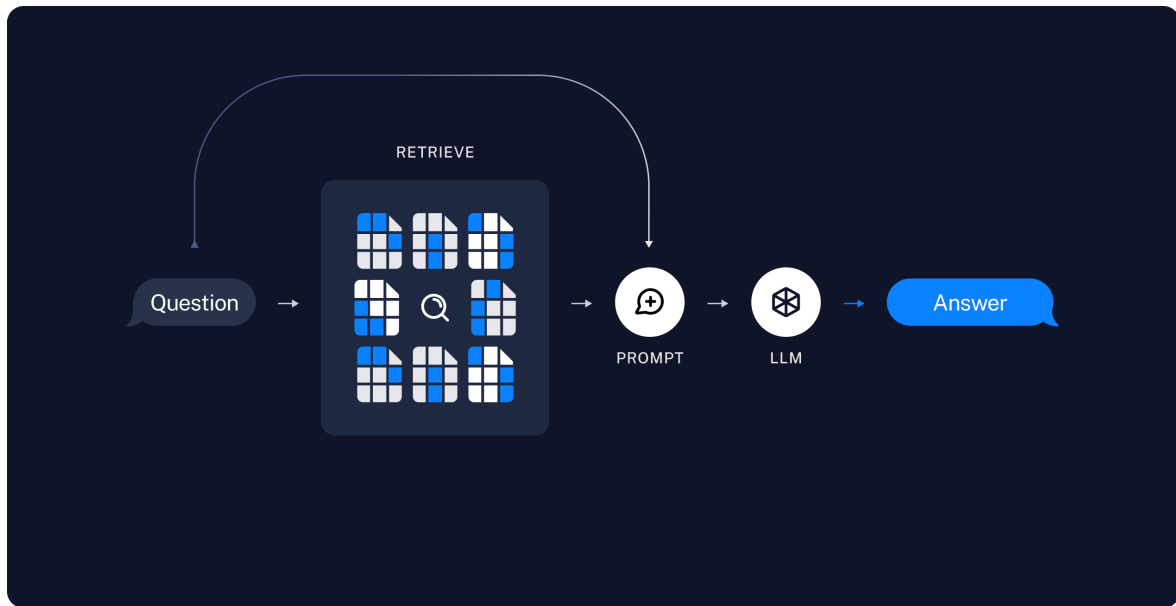


Abbildung 3: Beispielgrafik für das RAG-Retrieval und -Generierung aus der Lang-Chain-Dokumentation.

Quelle: <https://github.com/langchain-ai/docs> (MIT License)

Copyright (c) 2025 LangChain. Voller Lizenztext im Anhang, Seite IV

Zur Laufzeit wird die Eingabe ebenfalls eingebettet und gegen den Index gesucht, um die Top-k semantisch nächsten Chunks als Belege zu erhalten. In der Originalarbeit fungiert dieser Index als „nicht-parametrischer Speicher“, auf den ein separater Retriever zugreift. [20]

Im Generierungsschritt erhält das Generator-Modell (LLM) die ursprüngliche Eingabe zusammen mit den zuvor gefundenen Belegen und formuliert die Antwort auf Basis dieser Informationen.[19], [20] Relevante Stellen werden typischerweise entweder paraphrasiert oder, falls gefordert, wörtlich zitiert — abhängig von Prompt-Instruktionen und Decoding-Strategie. Wird ein Ziel-Ausgabeformat vorgegeben (z.B. ein JSON-Schema oder ein kurzes Beispiel), wird es dem Prompt mitgegeben. Das Modell füllt die geforderten Felder entsprechend aus. Es werden zwei Betriebsarten unterschieden: Bei RAG-Sequence bleibt das Modell während der gesamten Antwort bei einem Beleg, bei RAG-Token darf es während der Generierung zwischen Belegen wechseln und so Informationen aus mehreren Quellen kombinieren. In beiden Modi ist die Ausgabe stärker an Quellen gebunden, was die Neigung zu Halluzinationen vermindert.[19], [20]

Aus diesen Bausteinen folgen für diese Arbeit wichtige Vorteile: Das LLM erhält sowohl die unstrukturierte Eingabe als auch das Ziel-Ausgabeformat direkt „im Prompt“ (in-context). Ein ressourcenintensives, umfangreiches Fine-Tuning ist damit nicht erforderlich. Gleichzeitig reduziert die Bindung an nachprüfbare Belege

E. Kontextbasierte Generierung

die Halluzinationsneigung. Der Wissensstand lässt sich durch Aktualisieren oder Austauschen des Index zeitnah aktualisieren, ohne das Modell zu verändern.[19], [21]

F. Datenformate

III. Analyse und Anforderungsdefinition

A. Anforderungen der Polizei an einen KI-basierten Arbeitsablauf zur Vorgangserfassung

B. Funktionale Anforderungen

C. Nicht-funktionale Anforderungen

IV. Konzeption

A. Architekturentwurf

B. Datenmodellierung

C. Einbindung des LLMs in @rtus

D. Festlegung der Evaluationskriterien

V. Implementierung

A. Technologiewahl

B. Integration des LLMs in das @rtus Backend

C. Anbindung an vorhandene Datenstrukturen

VI. Evaluation und Tests

A. Testumgebung und Methodik

B. User-Studie: UI/UX

C. Ergebnisse

VII. Diskussion

A. Bewertung der Ergebnisse anhand der Hypothese

B. Grenzen des Systems

C. Praktische Relevanz für den polizeilichen Arbeitsalltag

VIII. Fazit und Ausblick

A. Zusammenfassung der Ergebnisse

B. Beantwortung der Forschungsfragen

C. Weiterführende Ansätze und mögliche Weiterentwicklungen

Literaturverzeichnis

- [1] X. Xing und P. Chen, „Entity Extraction of Key Elements in 110 Police Reports Based on Large Language Models“, *Applied Sciences*, Bd. 14, Nr. 17, S. 7819, Sep. 2024, doi: 10.3390/app14177819.
- [2] X. Chen, X. Tong, H. Lin, und Y. Xing, „Entity Recognition Method for Key Information of Police Records Based on Bert-Bilstm-Selfatt-Crf“, *Academic Journal of Computing & Information Science*, Bd. 7, Nr. 1, 2024, doi: 10.25236/AJCIS.2024.070112.
- [3] Y. Liu, Q. Guo, C. Yang, und Y. Liao, „TIPS: Tailored Information Extraction in Public Security Using Domain-Enhanced Large Language Model“, *Computers, Materials & Continua*, Bd. 83, Nr. 2, S. 2555–2572, 2025, doi: 10.32604/cmc.2025.060318.
- [4] C. H. Ku, A. Iriberry, und G. Leroy, „Crime Information Extraction from Police and Witness Narrative Reports“, in *2008 IEEE Conference on Technologies for Homeland Security*, Waltham, MA, USA: IEEE, Mai 2008, S. 193–198. doi: 10.1109/THS.2008.4534448.
- [5] C. H. Ku, A. Iriberry, und G. Leroy, „Natural Language Processing and E-Government: Crime Information Extraction from Heterogeneous Data Sources“, in *Proceedings of the 2008 International Conference on Digital Government Research*, in Dg.o '08. Montreal, Canada: Digital Government Society of North America, Mai 2008, S. 162–170.
- [6] J. Merilehto, „From PDFs to Structured Data: Utilizing LLM Analysis in Sports Database Management [Preprint]“. Zugegriffen: 16. September 2025. [Online]. Verfügbar unter: <http://arxiv.org/abs/2410.17619>
- [7] R. Peng, K. Liu, P. Yang, Z. Yuan, und S. Li, „Embedding-Based Retrieval with LLM for Effective Agriculture Information Extracting from Unstructured Data [Preprint]“. Zugegriffen: 16. September 2025. [Online]. Verfügbar unter: <http://arxiv.org/abs/2308.03107>
- [8] I. C. Wiest u. a., „LLM-AIx: An Open Source Pipeline for Information Extraction from Unstructured Medical Text Based on Privacy Preserving Large Language Models [Preprint]“. Zugegriffen: 29. September 2025. [Online]. Verfügbar unter: <http://medrxiv.org/lookup/doi/10.1101/2024.09.02.24312917>
- [9] C. Haas, „Technische Integration großer Sprachmodelle in Unternehmen: Entwicklung eines Frameworks zur technischen Vorbereitung von Unternehmen auf die Nutzung von Sprachmodellen“, S. 1494 KB, vi, 92 pages, 2025, doi: 10.58023/1155.

- [10] X. Li, S. Wang, S. Zeng, Y. Wu, und Y. Yang, „A Survey on LLM-based Multi-Agent Systems: Workflow, Infrastructure, and Challenges“, *Vicinagearth*, Bd. 1, Nr. 1, S. 9, Okt. 2024, doi: 10.1007/s44336-024-00009-2.
- [11] Z. Gero u. a., „Self-Verification Improves Few-Shot Clinical Information Extraction“, 2023, doi: 10.48550/ARXIV.2306.00024.
- [12] A. Vaswani u. a., „Attention Is All You Need“, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Zugegriffen: 2. Oktober 2025. [Online]. Verfügbar unter: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- [13] „Was sind Large Language Models (LLMs)? | IBM“. Zugegriffen: 2. Oktober 2025. [Online]. Verfügbar unter: <https://www.ibm.com/de-de/think/topics/large-language-models>
- [14] D. J. S. Kelbert Patricia, „Wie funktionieren LLMs? Ein Blick ins Innere großer Sprachmodelle - Blog des Fraunhofer IESE“. Zugegriffen: 2. Oktober 2025. [Online]. Verfügbar unter: <https://www.iese.fraunhofer.de/blog/wie-funktionieren-llms/>
- [15] „Introduction to Large Language Models | Machine Learning“. Zugegriffen: 2. Oktober 2025. [Online]. Verfügbar unter: <https://developers.google.com/machine-learning/resources/intro-llms>
- [16] „Decoding Strategies in Large Language Models“. Zugegriffen: 4. Oktober 2025. [Online]. Verfügbar unter: <https://huggingface.co/blog/mlabonne/decoding-strategies>
- [17] H. Naveed u. a., „A Comprehensive Overview of Large Language Models“, *ACM Transactions on Intelligent Systems and Technology*, Bd. 16, Nr. 5, S. 1–72, Okt. 2025, doi: 10.1145/3744746.
- [18] O. Ayala und P. Bechard, „Reducing Hallucination in Structured Outputs via Retrieval-Augmented Generation“, in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, Y. Yang, A. Davani, A. Sil, und A. Kumar, Hrsg., Mexico City, Mexico: Association for Computational Linguistics, Juni 2024, S. 228–238. doi: 10.18653/v1/2024.naacl-industry.19.
- [19] „What Is Retrieval-Augmented Generation (RAG)?“. Zugegriffen: 6. Oktober 2025. [Online]. Verfügbar unter: <https://research.ibm.com/blog/retrieval-augmented-generation-RAG>
- [20] P. Lewis u. a., „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks“, in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, und H. Lin, Hrsg., Curran Associates, Inc., 2020, S. 9459–9474. [Online]. Verfügbar unter: <https://proceedings.neurips>

cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

- [21] P. T. H. Kelbert Dr Julien Siebert, „Retrieval Augmented Generation (RAG): Chat mit eigenen Daten“. Zugegriffen: 6. Oktober 2025. [Online]. Verfügbar unter: <https://www.iese.fraunhofer.de/blog/retrieval-augmented-generation-rag/>

Anhang

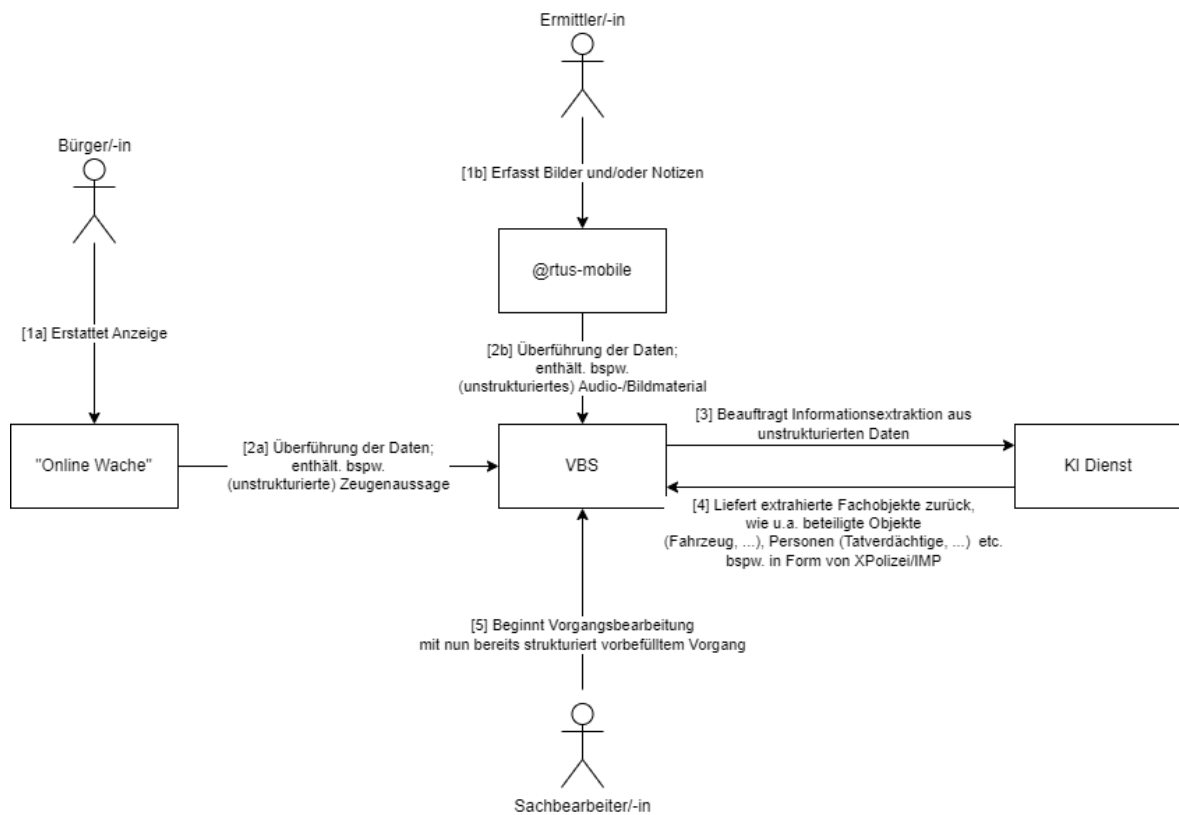


Abbildung 4: Datenfluss- und Verarbeitungsdiagramm: Anzeigen von Bürger/-innen (Onlinewache) und Erfassungen von Ermittler/-innen (@rtus-Mobile) liefern unstrukturierte Daten an das zentrale VBS, das einen KI-Dienst zur Informationsextraktion beauftragt; der KI-Dienst liefert strukturierte Fachobjekte (z. B. Personen, Fahrzeuge) zurück, woraufhin die Sachbearbeitung den vorbefüllten, strukturierten Vorgang weiterbearbeitet.