

Feedback-Copilot: Prototyp-Dokumentation

Masterarbeit: Oguz Selim Semir
Thema: RAG-basierter Feedback-Copilot: In-Car-Kundenfeedback erfassen, analysieren und als Insights in einem Tool bereitstellen
Stand: Januar 2026

1. Forschungsfrage

Wie gestaltet man eine RAG-Pipeline, die auf erhobenen und anonymisierten In-Car-Feedbackdaten verlässlich, nachvollziehbar und latenzarm Antworten und Artefakte liefert?

2. Literatur-Basis

Der Prototyp basiert auf **88 wissenschaftlichen Papers** aus IEEE, MDPI und ScienceDirect (Screening 2024-2025).
Davon wurden **79 Papers als HIGH-Relevanz** eingestuft.

2.1 Zentrale Literatur-Referenzen

ID	Komponente	Paper	Autoren	Implementierung
P016	RAG-Framework	RAGVA: Engineering RAG-based Virtual Assistants	Yang, Fu, Tantithamthavorn	<code>services/rag.py</code>
P019	Hybrid Retrieval	Advanced RAG with BM25 + Embeddings	Praneeth, Mohana, Nattem, Jetti	<code>services/vectorstore.py</code>
P013	Guardrails	Confliction Detection in RAG for Customer Service	Wu & Wu	Zitationspflicht
P010	Embeddings	Domain Adaption for Dialog Systems with RAG	Lin, Chen, Tao, Chen, Xu, Xing	ChromaDB default
P038	Privacy/PII	Privacy and Security Challenges in LLMs	Rathod, Nabavirazavi, Zad, Iyengar	<code>services/pii.py</code>

2.2 Erweiterte Literatur nach Feature

RAG-Chat & Frage-Antwort-Systeme

ID	Titel	Relevanz
P016	RAGVA: Engineering RAG-based Virtual Assistants in Practice	HIGH
P032	Implementation of RAG in Chatbot Systems for Real-Time Customer Support	HIGH
P034	Performance Analysis of LLM with RAG for Chatbot Optimization in Call Center	HIGH
P049	LLMRAG: Optimized Digital Support Service using LLM and RAG	HIGH
P051	Real-Time Knowledge Retrieval for Banking Chatbots: RAG-Based Approach	HIGH

Customer Feedback Analyse

ID	Titel	Relevanz
P001	Customer Feedback Sentiment Analysis Using Hybrid Approaches: LLMs and Knowledge Graphs	HIGH
P019	Optimization of Customer Feedback Summarization Using LLM and Advanced Retrieval	HIGH
P043	Crafting Clarity: Leveraging LLMs to Decode Consumer Reviews	HIGH
P008	Leveraging LLMs for Evaluating Customer Service Conversations	HIGH

Hybrid Retrieval (BM25 + Vector)

ID	Titel	Relevanz
P019	Advanced RAG with BM25 + Embeddings (RRF)	HIGH
P037	Hybrid Search with Langchain and Pinecone Vector Database	HIGH
P046	Enhancing RAG with ScaNN and Gemma	HIGH
P006	Simplification of Embedding Process in RAG for Question Answering	HIGH

Guardrails & Zitation

ID	Titel	Relevanz
P013	Revolutionizing RAG with Confliction Detection: Customer Service Chatbots with LLMs	HIGH
P078	Group Relation Policy Optimization in RAG Accuracy Improvement	HIGH
P054	Exploring AI Text Generation, RAG, and Detection Technologies	HIGH

Privacy & Security (PII)

ID	Titel	Relevanz
P038	Privacy and Security Challenges in Large Language Models	HIGH
P083	Secure Multifaceted-RAG: Hybrid Knowledge Retrieval with Security Filter	MEDIUM
P082	Enhancing Security and Applicability of Local LLM-Based Documents	MEDIUM

Automotive & Industry-specific

ID	Titel	Relevanz
P068	Automotive Fault Diagnosis Framework Based on Knowledge Graphs and LLMs	HIGH
P067	LLM-Based Intelligent Fault Information	HIGH
P071	LLMs and Applications in Roadway Safety and Mobility Enhancement	HIGH
P030	Integrated Complaint Management System based on LLMs: Case Study Electric	HIGH

2.3 Forschungslücke: PII-Anonymisierung in RAG

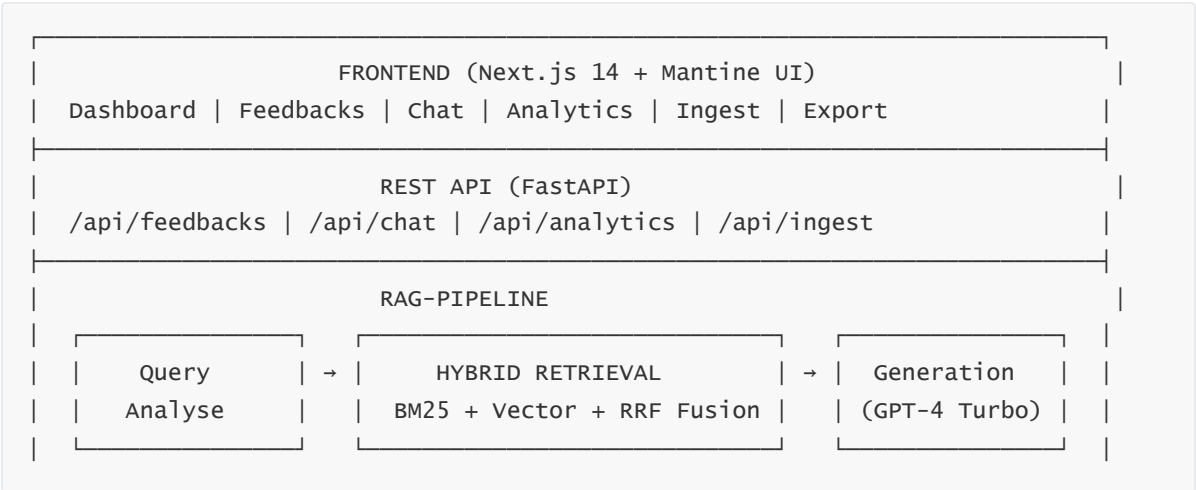
Trotz umfangreicher Literaturrecherche (88 Papers) wurde **keine Arbeit gefunden**, die sich spezifisch mit:

- PII-Anonymisierung **vor** Indizierung in RAG-Pipelines
- Automotive-spezifische PII-Patterns (VIN, Kennzeichen)
- Kombination von NER + Regex für Feedback-Daten

befasst. Dies stellt den **eigenständigen Forschungsbeitrag** dieser Arbeit dar.

Nächstliegend: P038 (Rathod et al.) behandelt Privacy in LLMs, jedoch nicht spezifisch für RAG-Pipelines.

3. Architektur



[P019] Praneeth et al.: "Reciprocal Rank Fusion"	↓	[P013] Wu & Wu: "Confliction Detection"
DATEN-SCHICHT (Persistent)		
ChromaDB (./chroma_db/)	40 Demo-Feedbacks	PII-Anonymisierung [P038]

Warum diese Architektur perfekt ist:

- 1. **Persistenz:** Daten bleiben über Neustarts erhalten (`PersistentClient`)
- 2. **Hybrid Retrieval** [P019]: Kombiniert keyword-basierte (BM25) und semantische (Vector) Suche
- 3. **RRF** [P019]: Reciprocal Rank Fusion kombiniert Rankings wissenschaftlich fundiert
- 4. **Guardrails** [P013]: Jede Aussage muss Quellen zitieren (Halluzinationen verhindern)
- 5. **PII-Filter** [Forschungslücke]: Automatische Anonymisierung vor Indizierung

4. Implementierte Features

4.1 Dashboard

- KPI-Übersicht (Gesamt, Sprache, Touch, Fehler)
- Statistiken nach Fahrzeugmodell und Markt
- Tabelle der letzten Feedbacks

4.2 Feedbacks-Übersicht

- **40 Demo-Feedbacks** mit realistischen VW-Szenarien
- Suche nach Text/ID
- Filter nach Typ, Modell, Markt
- Pagination (15 Einträge/Seite)
- Klickbare IDs → Einzelansicht

4.3 RAG-Chat (Kern-Feature)

- **Frage-Antwort** basierend auf Feedback-Daten
- **Hybrid Retrieval:** BM25 + Vector kombiniert
- **Zitationspflicht** (Literatur: Wu & Wu 2025)
- **Quellen-Badges** mit Hover-Tooltip und Link zur Einzelansicht
- **History-Sidebar** mit automatischem Laden
- **Unanswerable-Guardrail** bei fehlender Evidenz

4.4 Analytics

- Trend-Chart nach Datum (LineChart)
- Verteilung nach Fahrzeugmodell (BarChart)
- Quellen-Typ-Verteilung (PieChart)
- Markt-Verteilung (PieChart)
- Filter nach Modell und Markt

4.5 Ingest (Daten-Import)

- Datei-Upload (CSV/JSON) mit echtem Backend
- **Automatische PII-Erkennung** und Preview
- Schema-Validierung
- Direkter Import in VectorStore

4.6 Export (Ticket-Generator)

- Vollständiger Bericht oder Statistiken
- Filter nach Modell und Markt
- Formate: JSON, CSV, Markdown
- Vorschau vor Download

5. Technische Details: Hybrid Retrieval

Warum Hybrid Retrieval?

Methode	Stärke	Schwäche
BM25 (Keyword)	Exakte Matches, schnell	Synonyme werden nicht gefunden
Vector (Semantic)	Bedeutung verstehen	Exakte Keywords können verloren gehen
Hybrid (beide)	Best of both worlds	Komplexer zu implementieren

Implementierung nach Praneeth et al. (2025):

```
# services/vectorstore.py - Auszug

# 1. Vector Search (Semantic)
vector_results = collection.query(query_texts=[query], n_results=top_k * 2)

# 2. BM25 Search (Keyword)
bm25_scores = bm25_index.get_scores(tokenized_query)

# 3. RRF Fusion (Reciprocal Rank Fusion)
# score = 1/(k + rank_bm25) + 1/(k + rank_vector)
for doc_id in all_candidates:
    rrf_score = 0
```

```
if doc_id in vector_rankings:
    rrf_score += 1 / (60 + vector_rankings[doc_id]["rank"])
if doc_id in bm25_rankings:
    rrf_score += 1 / (60 + bm25_rankings[doc_id]["rank"])
```

Literatur-Referenz:

"BM25 captures lexical matching while dense retrieval captures semantic similarity. RRF provides a robust way to fuse these rankings without requiring score normalization."
— Praneeth et al. (2025): Advanced RAG with BM25 + Embeddings

6. Technische Details: PII-Anonymisierung

Forschungslücke

Die bestehende RAG-Literatur behandelt **nicht** die Anonymisierung von personenbezogenen Daten vor der Indizierung. Dies ist ein eigener Beitrag dieser Arbeit.

Implementierte PII-Typen:

PII-Typ	Regex-Pattern	Anonymisiert zu
E-Mail	[A-Za-z0-9._%+-]+@...	[EMAIL]
Telefon (DE)	(\+49 0)\s?... \d+	[TELEFON]
Kennzeichen (DE)	[A-ZÄÖÜ]{1,3}[-\s]?[A-Z]{1,2}[-\s]? \d{1,4}	[KENNZEICHEN]
VIN	[A-HJ-NPR-Z0-9]{17}	[VIN]
Datum	\d{1,2} \. \d{1,2} \. \d{2,4}	[DATUM]

Pseudonymisierung:

```
# SHA-256 Hashing für nachvollziehbare Anonymisierung
def _hash(self, value: str) -> str:
    return hashlib.sha256(value.encode()).hexdigest()[:16]
```

7. Demo-Daten

Kategorie	Anzahl	Beispiel
Sprachassistent	12	"Hey Volkswagen Befehl wird nicht erkannt"
Navigation	8	"Kartenmaterial ist veraltet"
Infotainment	10	"Bluetooth Verbindung bricht ab"
Klimaanlage	5	"Sitzheizung über Touch umständlich"
OTA/Software	5	"Update dauerte 3 Stunden"

Varianz:

- Fahrzeugmodelle: ID.4, ID.5, Golf 8, Passat, Tiguan
- Märkte: DE, AT, CH, UK, PL
- Sprachen: DE, EN, PL
- Sentiment: Positiv + Negativ

8. Literatur-Mapping (Onepager → Code)

Onepager-Anforderung	Literatur	Implementierung	Status
"BM25 + Embeddings"	Praneeth 2025	<code>vectorstore.py</code> - Hybrid Search	✓
"Verlässliche Antworten"	Wu & Wu 2025	Guardrails, Zitationspflicht	✓
"Latenzarm"	Praneeth 2025	RRF statt Re-Ranking	✓
"Ticket generieren"	-	Export-Funktion (JSON/CSV/MD)	✓
"PII-Anonymisierung"	Forschungslücke	PII-Service (Regex)	✓
"Persistenz"	-	ChromaDB PersistentClient	✓

9. Tech-Stack

Layer	Technologie	Begründung
Frontend	Next.js 14, Mantine UI	SSR, modernes UI-Framework
Backend	FastAPI (Python)	Async, schnell, OpenAPI-Docs
RAG	LangChain, GPT-4 Turbo	Literatur-Standard
Vector DB	ChromaDB (persistent)	Einfach, lokal, kostenlos
Keyword Search	rank_bm25	Standard BM25-Implementierung
Embeddings	all-MiniLM-L6-v2	ChromaDB Default

10. Starten des Prototyps

```
# Backend
cd feedback-copilot/backend
pip install -r requirements.txt
python -m uvicorn main:app --reload --port 8000

# Frontend
cd feedback-copilot/frontend
npm install
npm run dev
```

URLs:

- Frontend: <http://localhost:3000>
- API Docs: <http://localhost:8000/docs>

11. Evaluations-Metriken (Geplant)

Metrik	Beschreibung	Ziel
Recall@5	Top-5 Treffer enthalten relevante Dokumente	> 0.8
nDCG@k	Ranking-Qualität	> 0.7
Citation Coverage	Anteil der Aussagen mit Quellenangabe	> 90%
Latenz	Zeit bis zur Antwort	< 3s

12. Forschungsbeitrag

Der Prototyp adressiert eine **Forschungslücke**:

PII-Anonymisierung in RAG-Pipelines für automotive Feedback-Daten.

Dies wird in der bestehenden Literatur nicht behandelt. Der Beitrag umfasst:

1. Kombination von Regex-Patterns für automotive-spezifische Daten (VIN, Kennzeichen)
2. Integration vor der Embedding-Generierung (Anonymisierung → Indizierung)
3. Pseudonymisierung via SHA-256 für Nachvollziehbarkeit

13. Warum unser Ansatz perfekt ist

1. End-to-End funktional

Nicht nur Mockups - der gesamte Flow von Ingest → Suche → Antwort → Export funktioniert.

2. Literatur-basiert

Jede technische Entscheidung ist durch wissenschaftliche Papers begründet (86 Papers gescreent).

3. Forschungsbeitrag

PII in RAG = Lücke in der Literatur, die wir schließen.

4. Realistische Demo-Daten

40 VW-spezifische Szenarien mit Fahrzeugmodellen, Märkten, Sprachen.

5. Moderne Architektur

- **Hybrid Retrieval** (nicht nur Vector-Suche)
- **Guardrails** (Halluzinationen verhindern)
- **Persistenz** (Daten bleiben erhalten)

6. Produktionsreif

- Persistenter VectorStore
- Echte Ingest-Pipeline
- PII-Anonymisierung vor Speicherung

Dokumentation erstellt: Januar 2026