Technische Hochschule Nürnberg Georg Simon Ohm

Fakultät Informatik

Masterstudiengang Applied Research in Engineering Sciences

Personenbezogene Daten in Large Language Models

Erkennung, Maskierung und selektive Demaskierung von PII

Seminararbeit

im Modul *Large Language Models*

Verfasser: Maximilian Werzinger
Matrikelnummer: 3641088
E-Mail: werzingerma89360@th-nuernberg.de

Dozent: Prof. Dr. Florian Gallwitz
Abgabedatum: 19.01.2026

Wintersemester 2025/26

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Problemstellung

Large Language Models (LLMs) wie GPT-4, Claude oder LLaMA haben in den vergangenen Jahren bemerkenswerte Fortschritte erzielt und finden zunehmend Anwendung in Unternehmen und im Alltag. Diese Modelle werden auf riesigen Textkorpora trainiert, die häufig aus dem Internet stammen und unbeabsichtigt personenbezogene Daten (Personally Identifiable Information, PII) enthalten können. Carlini et al. [1] haben in ihrer Studie nachgewiesen, dass LLMs Trainingsdaten memorisieren und unter bestimmten Bedingungen wieder ausgeben können – inklusive sensibler Informationen wie Namen, E-Mail-Adressen und Telefonnummern.

Diese Problematik gewinnt vor dem Hintergrund der Datenschutz-Grundverordnung (DSGVO) besondere Bedeutung, da personenbezogene Daten einem besonderen Schutz unterliegen [2]. Die vorliegende Arbeit untersucht daher Methoden zur Erkennung und Maskierung von PII vor der Verarbeitung durch LLMs.

1.2 Zielsetzung der Arbeit

Die Arbeit verfolgt drei zentrale Ziele: Erstens soll das Problem personenbezogener Daten in LLM-Trainingsdaten und deren potenzielle Rekonstruktion erläutert werden. Zweitens wird ein praktisches System zur Erkennung und Maskierung von PII entwickelt und evaluiert. Drittens wird die Auswirkung der Maskierung auf die Qualität von LLM-Antworten im Retrieval-Augmented Generation (RAG) Setting untersucht.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich wie folgt: Kapitel 2 führt in die theoretischen Grundlagen ein und erläutert das Problem der Datenextraktion aus LLMs. Kapitel 3 beschreibt die verwendeten Erkennungs- und Maskierungsmethoden. Die praktische Implementierung inklusive der Bronze-, Silver- und Gold-Level-Artefakte wird in Kapitel 4 vorgestellt. Kapitel 5 präsentiert die Evaluationsergebnisse, bevor Kapitel 6 diese diskutiert und Kapitel 7 mit einem Fazit schließt.

2 Theoretische Grundlagen

2.1 Personenbezogene Daten (PII)

Personenbezogene Daten (Personally Identifiable Information, PII) umfassen alle Informationen, die zur direkten oder indirekten Identifizierung einer natürlichen Person genutzt werden können [2]. Das NIST Privacy Framework [7] bietet einen systematischen Ansatz zur Kategorisierung und zum Risikomanagement solcher Daten. Tabelle 1 zeigt eine Kategorisierung nach Sensitivität.

Tabelle 1: Kategorisierung personenbezogener Daten nach Sensitivität

Kategorie	**Beispiele**	**Risikostufe**
Direkte Identifikatoren	Name, E-Mail-Adresse	Mittel
Kontaktdaten	Telefonnummer, Adresse	Mittel
Sensible Identifikatoren	Sozialversicherungsnummer, IBAN	Hoch
Biometrische Daten	Fingerabdruck, Gesichtserkennung	Sehr hoch

2.2 PII in LLM-Trainingsdaten

Moderne LLMs werden typischerweise auf umfangreichen Webdaten trainiert, die aus Quellen wie Common Crawl, Wikipedia oder Reddit stammen. Diese Daten enthalten trotz Filterungsbemühungen häufig personenbezogene Informationen – sei es aus Forenbeiträgen, E-Mail-Leaks oder öffentlichen Dokumenten. Das Problem wird dadurch verstärkt, dass die schiere Datenmenge (oft mehrere Billionen Token) eine vollständige manuelle Überprüfung unmöglich macht.

2.3 Training Data Extraction

Carlini et al. [1] demonstrierten, dass LLMs Teile ihrer Trainingsdaten memorisieren und unter bestimmten Prompts wieder ausgeben können. Die Studie zeigte, dass aus GPT-2 mit 1.5 Milliarden Parametern hunderte verbatim memorisierte Sequenzen extrahiert werden konnten, darunter Namen, E-Mail-Adressen und Telefonnummern realer Personen.

Die Extraktion erfolgt typischerweise durch:

- **Prefix-Prompting**: Das Modell wird mit einem bekannten Textanfang gefüttert und zur Vervollständigung aufgefordert.
- **Membership Inference**: Es wird getestet, ob bestimmte Daten Teil des Trainingskorpus waren.
- **Model Inversion**: Rekonstruktion von Trainingsdaten durch Optimierung des Inputs.

Diese Erkenntnisse unterstreichen die Notwendigkeit, PII vor der Verarbeitung durch LLMs zu maskieren – sowohl beim Training als auch bei der Inferenz in RAG-Systemen.

3 Methodik

3.1 Erkennungsverfahren

Für die Erkennung personenbezogener Daten existieren verschiedene Ansätze, die sich in regelbasierte und ML-basierte Methoden unterteilen lassen. Etablierte Frameworks wie Microsoft Presidio [3] kombinieren beide Ansätze. Diese Arbeit entwickelt ein eigenes hybrides System, das die Stärken beider Methoden vereint.

3.1.1 Regelbasierte Erkennung (Regex)

Reguläre Ausdrücke (Regex) ermöglichen die präzise Erkennung strukturierter PII-Typen wie E-Mail-Adressen, Telefonnummern oder IBANs. Listing 1 zeigt ein Beispiel-Pattern für E-Mail-Adressen.

```
EMAIL_PATTERN = r'\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\b'
```

Listing 1: Regex-Pattern für E-Mail-Erkennung

Der Vorteil regelbasierter Erkennung liegt in der hohen Präzision für klar definierte Formate und der Nachvollziehbarkeit. Nachteile sind die fehlende Kontextsensitivität und die Unfähigkeit, unstrukturierte PII wie Namen zu erkennen. Die vollständige Sammlung aller verwendeten Regex-Patterns findet sich in Anhang A.

3.1.2 Named Entity Recognition (NER)

Named Entity Recognition (NER) nutzt maschinelles Lernen zur Erkennung benannter Entitäten im Text. Für diese Arbeit wird das deutsche spaCy-Modell `de_core_news_lg` eingesetzt, das Personennamen (PER), Orte (LOC) und Organisationen (ORG) erkennen kann [4].

Im Gegensatz zu Regex kann NER auch kontextabhängige PII wie *"Herr Müller"* oder *"Dr. Schmidt"* identifizieren, neigt jedoch zu höheren False-Positive-Raten bei mehrdeutigen Begriffen.

3.1.3 Hybride Erkennung

Der hybride Ansatz kombiniert Regex und NER, um die Stärken beider Methoden zu nutzen: Regex für strukturierte Daten mit hoher Präzision, NER für unstrukturierte Entitäten. Überlappende Matches werden dedupliziert, wobei der Match mit höherer Konfidenz bevorzugt wird.

3.2 Maskierungsstrategien

Die erkannten PII werden durch Platzhalter ersetzt, die den Typ und eine eindeutige ID kodieren, z.B. `[EMAIL_1]` oder `[NAME_2]`. Optional wird ein SHA-256-Hash des Originalwerts gespeichert, um spätere Verifizierung oder selektive Demaskierung zu ermöglichen.

4 Praktische Implementierung

Die praktische Umsetzung erfolgt in drei Stufen (Bronze, Silver, Gold), die aufeinander aufbauen und zunehmend komplexere Funktionalität bieten.

4.1 Bronze: Regel-/NER-basierte Erkennung und Maskierung

Die Bronze-Implementierung umfasst ein wiederverwendbares Python-Modul (`pii_redactor.py`), das sowohl Regex- als auch NER-basierte Erkennung unterstützt. Das Modul bietet eine einheitliche API für Erkennung, Maskierung und Statistikerfassung.

Listing 2 zeigt die Anwendung des Redaktors auf einen Beispieltext mit anschließender Ausgabe des maskierten Textes und des Platzhalter-Mappings.

```python
result = redactor_full.redact(example_text, method="hybrid")

# Ausgabe maskierter Text
print(result.redacted_text)

# Mapping fuer Demaskierung
print("Platzhalter-Mapping (Hash -> fuer spaetere Demaskierung):")
for placeholder, hash_val in result.mapping.items():
    print(f"  {placeholder} -> {hash_val}")
```

Listing 2: PII-Maskierung mit hybridem Ansatz

Die Ausgabe zeigt den maskierten Text, in dem alle erkannten PIIs durch typisierte Platzhalter ersetzt wurden:

```
Sehr geehrter Herr [NAME_1],
vielen Dank fuer Ihre Anfrage vom [DATE_OF_BIRTH_1].
Wir haben Ihre Daten erhalten:
- E-Mail: [EMAIL_1]
- Telefon: +[PHONE_1]
- IBAN: [IBAN_1]
[...]

Platzhalter-Mapping (Auszug):
  [NAME_1] -> dddfab9b5b8a3601
  [EMAIL_1] -> dd432348e6c3373c
  [IBAN_1] -> de12be61ad13570e
```

Die Erkennung liefert für jeden PII-Fund detaillierte Informationen: den erkannten Text, den PII-Typ, die Position im Text, einen Konfidenz-Score und die verwendete Methode (Regex oder NER). Tabelle 2 zeigt ein exemplarisches Erkennungsergebnis.

Tabelle 2: Exemplarische PII-Erkennung in einem Beispieltext

Erkannter Text	**Typ**	**Konfidenz**	**Methode**
Max Mustermann	NAME	0.85	NER
max.mustermann@example.com	EMAIL	0.95	Regex
+49 170 1234567	PHONE	0.85	Regex
DE89 3704 0044 0532 0130 00	IBAN	0.95	Regex

4.2 Silver: Analyse von False Positives und False Negatives

Die Silver-Stufe evaluiert die Erkennungsqualität anhand von 50 annotierten Beispielen, die sowohl aus dem WikiANN-Datensatz [5] als auch aus synthetisch erstellten Edge-Cases stammen. Die synthetischen Beispiele umfassen besonders schwierige Fälle wie:

- **Lowercase Names**: Namen ohne Großbuchstaben (z.B. *"max mustermann"*)
- **Unusual Formats**: Ungewöhnliche E-Mail-Formate mit Subdomains
- **Ambiguous**: Mehrdeutige Begriffe (z.B. *"Paris"* als Name oder Stadt)
- **Obfuscated**: Verschleierte Daten (z.B. *"0170-XXXX-1234"*)

Tabelle 3 zeigt die Verteilung der synthetischen Testfälle nach Kategorien.

Tabelle 3: Synthetische Testfälle nach Kategorien

Kategorie	Anzahl	Beschreibung
lowercase_names	3	Namen ohne Großbuchstaben
unusual_formats	4	Ungewöhnliche E-Mail-/Datumsformate
ambiguous	4	Mehrdeutige Begriffe (Paris, Tim)
obfuscated	3	Verschleierte Daten
multi_pii	2	Mehrere PIIs in einem Satz
false_positive_trap	3	Texte ohne echte PIIs
international	3	Internationale Formate
special_chars	3	Sonderzeichen in Namen

Zusätzlich wurden 25 Beispiele aus dem WikiANN-Datensatz [5] verwendet, sodass der Evaluationsdatensatz insgesamt 50 Beispiele mit 61 erwarteten PIIs umfasst.

Die Analyse zeigt, dass der hybride Ansatz die beste Balance zwischen Präzision und Recall bietet (siehe Tabelle 4).

Tabelle 4: Evaluationsergebnisse auf 50 Beispielen (61 erwartete PIIs)

Methode	Precision	Recall	F1-Score
Regex	0.765	0.206	0.325
Hybrid (Regex + NER)	0.761	0.818	0.788

Der hybride Ansatz erreicht einen deutlich höheren Recall (0.818 vs. 0.206), da NER auch unstrukturierte Entitäten wie Personennamen erkennt. Die leicht niedrigere Precision resultiert aus zusätzlichen False Positives bei mehrdeutigen Begriffen.

Die Analyse der 17 False Positives zeigt typische Problemmuster (siehe Tabelle 5).

Tabelle 5: Häufigste False Positives im Hybrid-Ansatz

Falsch erkannt	**Anzahl**	**Ursache**
WEITERLEITUNG	2	Großbuchstaben → Name
Geb.-Datum	1	Abkürzung → Name
Meeting	1	Englischer Begriff → Name
UK-Team	1	Ländercode → Name
König-Apotheke	1	Zusammengesetztes Wort → Name
192.168.1.1	1	Interne IP → PII
123.45	1	Preis → IP-Adresse

Die häufigsten False Negatives treten in den Kategorien `lowercase_names` und `ambiguous` auf, wo das NER-Modell ohne Großbuchstaben oder bei mehrdeutigen Begriffen versagt.

4.3 Gold: Selektive Demaskierung mit Audit-Log

Die Gold-Stufe implementiert ein vollständiges Zugriffsmanagement-System mit rollenbasierter Zugriffskontrolle (Role-Based Access Control, RBAC) und Audit-Logging. Das `audit_logger.py`-Modul definiert fünf Zugriffsebenen, die in Tabelle 6 dargestellt sind. Die Zugriffsebenen sind kumulativ aufgebaut, d.h. höhere Ebenen erben die Berechtigungen aller niedrigeren Ebenen (gekennzeichnet durch "+").

Tabelle 6: Zugriffsebenen und PII-Berechtigungen (kumulativ)

Level	**Beispielrolle**	**Zugriff auf**
PUBLIC	Externer Nutzer	Keine PII
INTERNAL	Mitarbeiter	Namen, E-Mails
CONFIDENTIAL	HR-Manager	+ Telefon, Adresse, Geburtsdatum
RESTRICTED	Datenschutzbeauftragter	+ SSN, Kreditkarten, IBAN
ADMIN	System-Admin	Vollzugriff

Jeder Demaskierungszugriff wird protokolliert und enthält:

- Zeitstempel und Benutzerinformationen
- Angefragten PII-Typ und Platzhalter
- Erfolg/Ablehnung mit Begründung
- Hash des Originalwerts für Nachvollziehbarkeit

Listing 3 zeigt die Registrierung der erkannten PIIs im Audit-System:

```
# PIIs im Audit-Logger registrieren
for match in redaction_result.matches:
    hash_value = audit_logger.register_pii(
        placeholder=match.placeholder,
```

```
        original=match.text,
        pii_type=match.pii_type.value
    )
    print(f"  {match.placeholder} -> {match.pii_type.value} (Hash: {
    hash_value})")
print(f"\n{len(redaction_result.matches)} PIIs gefunden")
```

Listing 3: Registrierung von PIIs im Audit-Logger

```
  [NAME_1] -> name (Hash: dd825657c9fe9c4f)
  [EMAIL_1] -> email (Hash: 8e401fd516e1709a)
  [PHONE_1] -> phone (Hash: 150ec31d02352af3)
  [IBAN_1] -> iban (Hash: de12be61ad13570e)
  [...]
12 PIIs gefunden
```

Die Integration mit einem lokalen LLM (LLaMA 3.2) zeigt, dass verschiedene Benutzer bei identischen Anfragen unterschiedliche Antworten erhalten – abhängig von ihren Zugriffsrechten.

Listing 4 demonstriert verschiedene Zugriffsszenarien:

```
# 1. Public User versucht E-Mail zu sehen
demonstrate_demask_attempt(
    user=test_users['public'],
    placeholder="[EMAIL_1]",
    reason="Kunde moechte Kontaktdaten"
)

# 2. HR-Manager sieht Telefonnummer
demonstrate_demask_attempt(
    user=test_users['confidential'],
    placeholder="[PHONE_1]",
    reason="Dringender Rueckruf erforderlich"
)
```

Listing 4: Demaskierungs-Szenarien mit unterschiedlichen Berechtigungen

Die Ausgabe zeigt die unterschiedlichen Ergebnisse je nach Zugriffsebene:

```
[SZENARIO 1] Externer Nutzer versucht E-Mail zu sehen:
Benutzer: guest_user (PUBLIC)
Angefordert: [EMAIL_1]
ERGEBNIS:
  Aktion: DEMASK_DENIED
  Grund: Zugriff verweigert: PUBLIC < INTERNAL

[SZENARIO 2] HR-Manager sieht Telefonnummer:
Benutzer: hr_manager (CONFIDENTIAL)
Angefordert: [PHONE_1]
ERGEBNIS:
  Aktion: DEMASK_SUCCESS
  Demaskierter Wert: 49 89 123456789
```

Detaillierte Beispiele für Audit-Log-Einträge im JSON-Format finden sich in Anhang B.

5 Evaluation

5.1 Qualitätsmetriken

Die Evaluation der Maskierung erfolgt anhand mehrerer Metriken:

- **ROUGE-Scores**: Messung der Wortüberlappung zwischen Original- und maskierter Antwort. ROUGE-1 betrachtet einzelne Wörter (Unigramme), ROUGE-L die längste gemeinsame Teilsequenz.
- **Semantische Ähnlichkeit**: Die Cosine-Ähnlichkeit misst den Winkel zwischen zwei Vektoren im hochdimensionalen Raum. Sentence-Embeddings sind dabei numerische Repräsentationen ganzer Sätze, die von vortrainierten Modellen erzeugt werden. Ein Wert von 1.0 bedeutet identische Semantik, 0.0 keine Ähnlichkeit.
- **Leakage-Rate**: Anteil der PII, die trotz Maskierung in der LLM-Antwort erscheinen.

5.2 Ergebnisse

Tabelle 7: Qualitätsvergleich: Original vs. maskierte LLM-Antworten

Anfrage	ROUGE-1	ROUGE-L	Leak. (Orig.)	Leak. (Mask.)	Sem. Ähnl.
Welche Kontaktdaten hat der Kunde?	0.312	0.312	0.0%	0.0%	0.552
Was ist die Beurteilung der Mitarbeiterin?	0.098	0.098	33.3%	0.0%	0.000

Die Ergebnisse zeigen, dass die Maskierung die Leakage-Rate signifikant reduziert (von 33.3% auf 0.0%), während die semantische Qualität der Antworten weitgehend erhalten bleibt. Die semantische Ähnlichkeit von 0.000 bei der zweiten Anfrage erklärt sich dadurch, dass die maskierte Antwort ohne den konkreten Personennamen eine fundamental andere Struktur aufweist – das Modell kann keine spezifische Beurteilung zu `[NAME_1]` liefern und antwortet stattdessen allgemein. Dies ist ein erwartetes Verhalten, das den Schutz personenbezogener Daten bestätigt.

5.3 Vergleich der Erkennungsmethoden

Tabelle 8 fasst die Ergebnisse der verschiedenen Erkennungsmethoden zusammen. Für NER allein wurden keine separaten Precision- und Recall-Werte erhoben, da der Fokus auf dem Vergleich zwischen reinem Regex und dem hybriden Ansatz lag.

Tabelle 8: Zusammenfassender Vergleich der Erkennungsmethoden

Methode	**Precision**	**Recall**	**F1-Score**	**Avg. Zeit**
Regex	0.765	0.206	0.325	0.03 ms
NER (spaCy)	–	–	0.640	7.24 ms
Hybrid	0.761	0.818	0.788	∼7.3 ms

Der hybride Ansatz kombiniert die Stärken beider Methoden: Die hohe Präzision von Regex bei strukturierten Daten (E-Mails, IBANs) und den hohen Recall von NER bei unstrukturierten Entitäten (Namen). Die zusätzliche Verarbeitungszeit von ca. 7 ms ist für die meisten Anwendungsfälle akzeptabel.

6 Diskussion

6.1 Vorteile des entwickelten Systems

Das entwickelte System bietet mehrere Vorteile: Die modulare Architektur ermöglicht einfache Erweiterung um neue PII-Typen. Die hybride Erkennung kombiniert die Stärken von Regex (Präzision) und NER (Kontextsensitivität). Das RBAC-System mit Audit-Logging erfüllt Compliance-Anforderungen nach DSGVO.

6.2 Limitationen

Die Limitationen umfassen: Die NER-Erkennung ist sprachabhängig und erfordert separate Modelle für verschiedene Sprachen. False Positives bei mehrdeutigen Begriffen können zu Über-Maskierung führen. Die Performance sinkt bei sehr langen Dokumenten, da NER-Modelle kontextuell arbeiten.

6.3 Ausblick

Zukünftige Arbeiten könnten LLM-basierte PII-Erkennung untersuchen, die Kontextverständnis mit höherer Genauigkeit verbindet. Auch die Integration von Differential Privacy [6] beim LLM-Training könnte das Problem an der Wurzel adressieren.

7 Fazit

Diese Arbeit hat das Problem personenbezogener Daten in Large Language Models untersucht und ein praktisches System zur Erkennung, Maskierung und selektiven Demaskierung entwickelt. Die Evaluation zeigt, dass der hybride Ansatz aus Regex und NER die beste Balance zwischen Präzision und Recall bietet, während das rollenbasierte Zugriffsmanagement (Role-Based Access Control, RBAC) Compliance-Anforderungen erfüllt.

Die entwickelten Module (`pii_redactor.py` und `audit_logger.py`) sind wiederverwendbar und können in bestehende RAG-Pipelines integriert werden. Die Ergebnisse unterstreichen die Wichtigkeit von PII-Schutzmaßnahmen beim Einsatz von LLMs in datenschutzsensiblen Kontexten.

Literatur

[1] Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., & Raffel, C. (2021). *Extracting Training Data from Large Language Models*. 30th USENIX Security Symposium. https://arxiv.org/abs/2012.07805

[2] Europäisches Parlament und Rat der Europäischen Union (2016). *Verordnung (EU) 2016/679 (Datenschutz-Grundverordnung)*. https://eur-lex.europa.eu/eli/reg/2016/679/oj/deu

[3] Microsoft (2024). *Presidio - Data Protection and De-identification SDK*. GitHub Repository. https://github.com/microsoft/presidio

[4] Explosion AI (2024). *spaCy - Industrial-Strength Natural Language Processing*. https://spacy.io/

[5] Pan, X., Zhang, B., May, J., Nothman, J., Knight, K., & Ji, H. (2017). *Cross-lingual Name Tagging and Linking for 282 Languages*. Proceedings of ACL 2017.

[6] Dwork, C., & Roth, A. (2014). *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science, 9(3-4), 211-407.

[7] National Institute of Standards and Technology (2020). *NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management*. https://www.nist.gov/privacy-framework

A Regex-Patterns

Die folgenden Regex-Patterns werden im `pii_redactor.py`-Modul zur Erkennung strukturierter PII verwendet:

```
DEFAULT_PATTERNS = {
    PIIType.EMAIL: {
        # E-Mail: user@domain.tld
        "pattern": r'\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\b',
        "confidence": 0.95
    },
    PIIType.PHONE: {
        # Deutsche Telefonnummern mit optionaler Landesvorwahl
        "pattern": r'\b(?:\+?49[-.\s]?)?(?:\(?\d{2,5}\)?[-.\s]?)?\d{3,}[-.\s]?\d{2,}\b',
        "confidence": 0.85
    },
    PIIType.IBAN: {
        # IBAN: 2 Buchstaben + 2 Pruefziffern + bis zu 30 Zeichen
        "pattern": r'\b[A-Z]{2}\d{2}(?:\s?\d{4}){4,7}\d{0,2}\b',
        "confidence": 0.95
    },
    PIIType.SSN: {
        # Deutsche Sozialversicherungsnummer: 12 Zeichen
        "pattern": r'\b\d{2}\s?\d{6}\s?[A-Z]\s?\d{3}\b',
        "confidence": 0.90
    },
    PIIType.IP_ADDRESS: {
        # IPv4-Adressen: 4 Oktette (0-255)
        "pattern": r'\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b',
        "confidence": 0.95
    },
    PIIType.DATE_OF_BIRTH: {
        # Geburtsdaten: DD.MM.YYYY oder DD/MM/YYYY
        "pattern": r'\b(?:0[1-9]|[12][0-9]|3[01])[./](?:0[1-9]|1[0-2])[./](?:19|20)\d{2}\b',
        "confidence": 0.85
    }
}
```

Listing 5: Regex-Patterns für PII-Erkennung

B Audit-Log Beispiel

Das folgende JSON zeigt einen exemplarischen Audit-Log-Eintrag für einen abgelehnten Demaskierungsversuch:

```
{
  "entry_id": "74ebac5a-d2a0-4acd-a0d9-ced4288eeb4d",
  "timestamp": "2025-12-22T03:42:29.098521",
  "user": {
    "user_id": "u001",
    "username": "guest_user",
```

```
    "access_level": "PUBLIC",
    "department": "External",
    "roles": ["viewer"]
  },
  "action": "DEMASK_DENIED",
  "pii_type": "email",
  "placeholder": "[EMAIL_1]",
  "original_hash": "8e401fd516e1709a",
  "success": false,
  "reason": "Zugriff verweigert: PUBLIC < INTERNAL",
  "ip_address": "192.168.1.100",
  "session_id": "sess_6e1051c5"
}
```

Listing 6: Audit-Log Eintrag (DEMASK_DENIED)

Ein erfolgreicher Zugriff wird wie folgt protokolliert:

```
{
  "entry_id": "8d1d6d88-c3b6-4ba2-8ee9-f10db09e6937",
  "timestamp": "2025-12-22T03:42:29.099321",
  "user": {
    "user_id": "u005",
    "username": "system_admin",
    "access_level": "ADMIN",
    "department": "IT",
    "roles": ["admin", "superuser"]
  },
  "action": "DEMASK_SUCCESS",
  "pii_type": "iban",
  "placeholder": "[IBAN_1]",
  "original_hash": "de12be61ad13570e",
  "success": true,
  "reason": "Audit-Anforderung fuer Jahresabschluss, Ticket #2024-1234",
  "ip_address": "192.168.1.100",
  "session_id": "sess_a4683531"
}
```

Listing 7: Audit-Log Eintrag (DEMASK_SUCCESS)

Die Zusammenfassung eines Export-Logs enthält aggregierte Statistiken:

```
"summary": {
  "total_entries": 29,
  "successful_accesses": 24,
  "denied_accesses": 5,
  "unique_users": 3,
  "accesses_by_pii_type": {
    "email": 3, "phone": 5, "iban": 3,
    "name": 6, "date_of_birth": 2, "address": 4,
    "ssn": 2, "ip_address": 2
  },
  "accesses_by_action": {
    "DEMASK_DENIED": 3,
    "DEMASK_SUCCESS": 24,
    "DEMASK_ATTEMPT": 2
  }
}
```

Listing 8: Audit-Log Zusammenfassung