universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme (DBIS)

Identifikation von DSGVO-kritischen Aktivitäten in Business Prozessen mittels Large Language Models

Abschlussarbeit an der Universität Ulm

Vorgelegt von:
Merten Dieckmann
merten.dieckmann@uni-ulm.de
1058340

Gutachter:
Prof. Dr. Manfred Reichert
Prof. Dr. Rüdiger Pryss

Betreuer:
Magdalena von Schwerin

2025

Fassung 29. September 2025

Satz: PDF-LaTeX 2ε

Inhaltsverzeichnis

Abkürzungen

DSGVO Datenschutz-Grundverordnung
KI Künstliche Intelligenz
LM Language Model
LLM Large Language Model
RAG Retrieval Augmented Generation
BPMN Business Process Model and Notation
EU Europäische Union
TP True Positive
TN True Negative
FP False Positive
FN False Negative

1 Einleitung

1.1 Motivation

Geschäftsprozesse sind in nahezu allen Organisationen allgegenwärtig und bilden die Grundlage für effiziente Abläufe. Zugleich ist in Europa durch die Datenschutz-Grundverordnung (DSGVO) der Datenschutz zu einem zentralen regulatorischen Aspekt geworden [9, 11]. Unternehmen müssen sicherstellen, dass in ihren Prozessen personenbezogene Daten rechtskonform verarbeitet werden; andernfalls drohen Strafen von bis zu 20 Millionen Euro oder 4% des weltweit gesamten erzielten Jahresumsatzes [11].

Die Überprüfung von Prozessen auf Konformität in Bezug auf Datenschutz ist jedoch zeit- und kostenintensiv [22, 35]. Besonders in großen Organisationen mit hunderten parallel laufenden Prozessen ist eine manuelle Analyse kaum praktikabel und zudem fehleranfällig. Fehlerhafte Untererkennungen datenschutzkritischer Aktivitäten (False Negatives) können weitreichende Folgen haben – von Reputationsschäden bis hin zu hohen Bußgeldern [22].

Vor diesem Hintergrund rücken Large Language Models (LLMs) als aufstrebende Künstliche Intelligenz (KI) Technologie in den Fokus. Sie sind darauf trainiert, natürliche Sprache auch in langen und komplexen Texten zu verstehen, Zusammenhänge über große Kontexte hinweg zu erkennen und Anweisungen zu befolgen. Damit erscheinen LLMs als vielversprechender Ansatz für das automatisierte Screening von Prozessmodellen. Erste Arbeiten belegen dieses Potenzial, etwa bei der Identifikation datenschutzrelevanter Verarbeitungstätigkeiten oder in der Analyse von Datenschutzerklärungen [10, 28].

Besonders interessant sind in diesem Kontext europäische Open-Source-Modelle wie die von Mistral [2]. Sie sind zum einen frei verfügbar und transparent, zum anderen wurden sie bislang kaum im Hinblick auf DSGVO-bezogene Aufgaben evaluiert.

Es fehlen belastbare, reproduzierbare empirische Vergleiche, die eine fundierte Bewertung dieser Modelle erlauben würden [34].

1.2 Problemstellung

Trotz der genannten Potenziale fehlt es bisher an standardisierten, reproduzierbaren Vergleichen verschiedener Modelle für die konkrete Aufgabe Aktivitäten in Geschäftsprozessen nach „kritisch“ und „unkritisch“ zu klassifizieren. Erste Ansätze, wie z.B. der von Nake et al. [22], zeigen dass maschinelles Lernen grundsätzlich in der Lage ist DSGVO-kritische Aktivitäten in textuellen Prozessbeschreibungen zu erkennen; dennoch existieren keine einheitlichen Benchmarks, die einen systematischen vergleiche unterschiedlicher LLMs erlauben.

Auch von Schwerin et al. [34] heben hervor, dass trotz großer Fortschritte im Einsatz von LLMs für juristische Aufgaben bislang erhebliche Lücken in der Evaluation für compliance-spezifische Anwendungen bestehen und geeignete DSGVO-spezifische Benchmarks fehlen. Somit mangelt es derzeit an einer belastbaren empirischen Grundlage, um Modelle zuverlässig und vergleichbar zu bewerten.

Besonders interessant ist die Frage, wie sich Open-Source-Modelle - insbesondere mit Ursprung aus der Europäische Union (EU) - im Vergleich zu internationalen außerhalb der EU entwickelten Modellen schlagen und welche Trade-offs dabei entstehen [34]. Diese Perspektive ist nicht nur aus Leistungs-, sondern auch aus Transparenz- und Regulierungsgründen relevant.

Eine zusätzliche Herausforderung ergibt sich aus der Natur von Business Process Model and Notation (BPMN)-Modellen: Typischerweise konzentrieren sie sich auf den Kontrollfluss und vernachlässigen die Datenebene. Datenobjekte werden oftmals gar nicht explizit modelliert oder nur implizit in den Aktivitäten referenziert. Dadurch ist die Datennutzung von Aktivitäten nicht direkt erkennbar und muss aus textuellen Beschreibungen und dem Kontext erschlossen werden [32]. Das erschwert die automatische Identifikation von DSGVO-kritischen Aktivitäten, da Algorithmen personenbezogene Datenflüsse zunächst indirekt und über den Kontext ableiten müssen.

1.3 Zielsetzung und Beiträge

Ziel der Arbeit ist es, einen methodischen Beitrag zur automatisierten Identifikation von DSGVO-kritischen Aktivitäten in Geschäftsprozessen zu leisten. Hierfür werden folgende Beiträge angestrebt:

- Entwicklung einer Klassifizierungspipeline für Geschäftsprozesse, die Aktivitäten binär in datenschutzkritisch oder unkritisch einordnet.
- Konzeption und Umsetzung eines Evaluationsframeworks, das reproduzierbare Vergleiche verschiedener LLMs und Algorithmen über eine einheitliche Schnittstelle ermöglicht.
- Entwicklung einer Labelingsoftware zur Erstellung und Annotation von Datensätzen für das Evaluationsframework.
- Aufbau eines repräsentativen Datensatzes aus gelabelten BPMN-Prozessen, inklusive klar definierter Labeling-Kriterien.
- Bereitstellung überprüfbarer empirischer Befunde, inklusive Code, Konfigurationen der Experimente und Seeds, um Nachvollziehbarkeit und Reproduzierbarkeit zu gewährleisten.

1.4 Forschungsfrage und Unterfragen

Die zentrale Forschungsfrage dieser Arbeit lautet:

FF1: Wie zuverlässig identifizieren LLMs DSGVO-kritische Aktivitäten in BPMN-Prozessmodellen?

Um diese Frage differenziert beantworten zu können werden außerdem folgende Unterfragen betrachtet:

- *UF1: Wie gut schneiden europäische Open-Source-Modelle im Vergleich zu internationalen Modellen ab?*
- *UF2: Wie unterscheiden sich große und kleine Modelle in ihrer Leistungsfähigkeit?*

- *UF3: Welche Open-Source-Modelle (insbesondere aus der EU) erzielen die besten Ergebnisse?*
- *UF4: Wie gut schneiden Open-Source-Modelle gegenüber kommerziellen Modellen wie GPT-4o ab?*

Für ein initiales Screening reicht, wie in [22], eine binäre Klassifikation (kritisch vs. unkritisch). Eine tiefergehende rechtliche Prüfung kann in einem nachfolgendem Schritt durchgeführt werden und ist nicht Bestandteil dieser Arbeit.

1.5 Aufbau der Arbeit

Die Arbeit ist wie folgt gegliedert: Kapitel 2 gibt einen Überblick über den theoretischen Hintergrund, die DSGVO und BPMN sowie eine Einführung in LLMs und verwandte Arbeiten. Kapitel 3 beschreibt den Rahmen der Entwicklung der Klassifizierungspipeline, des Evaluationsframeworks und der Experimente. Kapitel 4 stellt den entwickelten Algorithmus zur Klassifikation von BPMN-Modellen und dessen einheitliche Schnittstelle vor. Kapitel 5 präsentiert die Architektur und den Funktionsumfang der Evaluationspipeline. Anschließend wird in Kapitel 6 die Labelingsoftware und die Erstellung der Datensätze erläutert. Kapitel 7 zeigt auf wie die Auswahl der LLMs erfolgte. Kapitel 8 erläutert den Versuchsaufbau, Kapitel 9 die Durchführung der Experimente und Kapitel 10 stellt die Ergebnisse vor. In Kapitel 11 werden die Ergebnisse im Kontext der Forschungsfragen diskutiert. Zum Schluss fasst Kapitel 12 die Arbeit zusammen und Kapitel 13 gibt einen Ausblick auf mögliche zukünftige Forschungsthemen.

2 Hintergrund und verwandte Arbeiten

2.1 Datenschutzgrundverordnung (DSGVO)

Die europäische Datenschutz-Grundverordnung (DSGVO) [11] bildet den zentralen rechtlichen Rahmen für den Schutz personenbezogener Daten in der EU. Sie gilt seit dem 25. Mai 2018. Durch die DSGVO werden Betroffenenrechte gestärkt und Verantwortliche zu technischen und organisatorischen Maßnahmen verpflichtet, wie z. B. *Datenschutz durch Technikgestaltung* und *datenschutzfreundliche Voreinstellungen* (Art. 25 DSGVO) [12].

Definitionen

Im Folgenden werden zentrale Begriffe der DSGVO erläutert, die für das Verständnis dieser Arbeit relevant sind:

- **Personenbezogene Daten** sind alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person beziehen (Art. 4 Abs. 1 DSGVO) [11]. Eine Person ist identifizierbar, wenn sie direkt oder indirekt bestimmbar ist (z. B. anhand von Name, Kennnummer, Standortdaten, Online-Kennung).
- **Verarbeitung** bezeichnet *jeden* mit personenbezogenen Daten vorgenommenen Vorgang (Art. 4 Abs. 2 DSGVO) und umfasst insbesondere das **Erheben**, **Speichern**, **Verwenden**/**Nutzen**, **Offenlegen durch Übermittlung** sowie das **Löschen**/**Vernichten** [11].
- Im BPMN-Kontext sind alle Aktivitäten als **datenschutzkritisch** zu betrachten, die solche Verarbeitungshandlungen an personenbezogenen Daten vor-

nehmen oder auslösen (z. B. Abruf aus einem Kundendaten-Speicher, Übergabe an externe Stellen).

Abgrenzung: Risiko-Screening vs. Rechtsberatung

Die in dieser Arbeit eingesetzten Klassifizierungsverfahren dienen einem *automatisierten Risiko-Vorscreening* von Prozessaktivitäten. Sie ersetzen keine individuelle Rechtsprüfung im Einzelfall. Insbesondere in Deutschland ist die Erbringung konkreter Rechtsdienstleistungen Personen mit entsprechender Befugnis vorbehalten [6]. Die Ergebnisse sind daher als Entscheidungshilfe zu verstehen und bedürfen - insbesondere bei Grenzfällen - der Bewertung durch qualifizierte Experten.

2.2 Business Process Model and Notation (BPMN)

BPMN ist ein Standard zur Modellierung von Geschäftsprozessen. Die Notation wurde entwickelt, um eine einheitliche Notation bereitzustellen, die sowohl von Geschäftsanalysten als auch von technischen Entwicklern verstanden wird. BPMN-Modelle bestehen aus verschiedenen Elementen wie Aktivitäten, Ereignissen, Gateways und Verbindungen, die zusammen den Ablauf eines Geschäftsprozesses darstellen [23].

Relevante BPMN-Elemente

Für die Identifikation von DSGVO-kritischen Aktivitäten sind insbesondere folgende Elemente relevant, da sie Hinweise auf den Umgang mit (personenbezogenen) Daten geben. Sie sind ebenfalls in Abbildung 2.1 dargestellt:

- **Aktivitäten** bilden die auszuführenden Arbeitsschritte eines Prozesses ab. Sie können Ein- und Ausgaben sowie Datenabhängigkeiten definieren [23]. Durch ihren Namen oder Kontext können Rückschlüsse auf die Verarbeitung personenbezogener Daten gezogen werden.

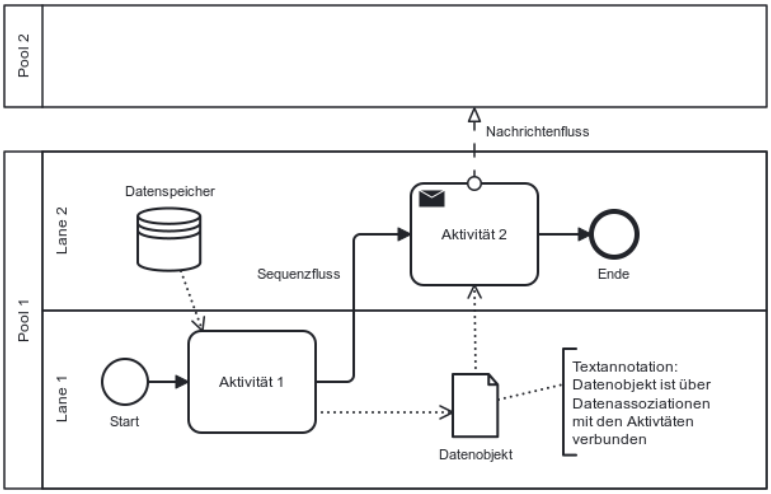
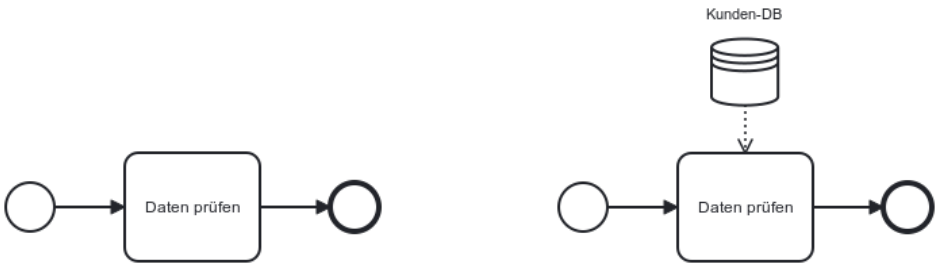


Abbildung 2.1: Die relevanten BPMN Elemente in Beziehungen zueinander

- **Sequenzflüsse** verbinden Aktivitäten, Ereignisse und Gateways und zeigen die Reihenfolge der Ausführung im Prozess an [23]. Mit ihrer Hilfe kann eine einzelne Aktivität im Kontext des gesamten Prozesses betrachtet werden, indem der Pfad zu und von der Aktivität verfolgt wird.
- **Datenobjekte und Datenspeicher** repräsentieren flüchtige oder persistente Daten, die im Prozess von z.B. Aktivitäten genutzt oder geschrieben werden können [23]. Sie können auch personenbezogene Daten enthalten.
- **Datenassoziationen** (Eingangs- und Ausgangsassoziationen) verbinden Aktivitäten mit Datenobjekten und Datenspeichern und zeigen so Ein- und Ausgaben explizit an [23]. Sie sind ein wichtiges Signal für die Verarbeitung personenbezogener Daten, da sie den direkten Bezug einer Aktivität zu bestimmten Daten verdeutlichen (z.B. Lesezugriff auf eine Kundendatenbank).
- **Pools** modellieren Organisationseinheiten oder Prozessbeteiligte, während **Lanes** Verantwortlichkeiten innerhalb eines Pools darstellen. Innerhalb eines Pools befinden sich die Aktivitäten und anderen Elemente des Prozesses [23]. Die Rollen und Verantwortlichkeiten, die durch Pools und Lanes dargestellt werden, können für die Bewertung der Datenverarbeitung relevant sein.
- **Nachrichtenflüsse** stellen den Austausch von Nachrichten zwischen verschiedenen Pools dar [23]. Sie können auf eine Übermittlung personenbezogener Daten an Dritte hinweisen (z.B. Transfer von Kundendaten an einen



(a) Ohne Datenassoziation (b) Mit Datenassoziation

Abbildung 2.2: Beispiel einer Datenassoziation als Datenschutzsignal.

externen Dienstleister).

- **Textannotationen und Assoziationen** dienen dazu, zusätzliche Informationen zu Prozessmodellen hinzuzufügen, die nicht durch die standardmäßigen BPMN-Elemente abgedeckt sind [23]. Sie können genutzt werden, um die Art der Datenverarbeitung zu präzisieren (z.B. „enthält E-Mail-Adresse“).

BPMN-XML

BPMN-Modelle werden in einer XML-Serialisierung gespeichert (BPMN 2.0 XML) [23]. Diese Darstellung enthält alle relevanten Strukturinformationen (Elementtypen, Namen, Beziehungen, Zuordnungen, Positionen der Elemente) und wird von vielen Prozess-Engines und Modellierungswerkzeugen wie Camunda [14] und BPMN.io [13] unterstützt. Für diese Arbeit dient BPMN-XML als Eingabeformat der Klassifizierungspipeline (siehe Kapitel 4).

Im Metamodell von BPMN erben fast alle Elemente von `BaseElement` und damit ein `id`-Attribut. Dieses `id` dient der eindeutigen Referenzierung und ist erforderlich [23]. Diese stabile `id` ist für die Klassifizierungspipeline wichtig, da sie eine stabile Referenzierung der Aktivitäten und anderer Elemente ermöglicht. Dies ist insbesondere dann relevant, wenn die Ergebnisse der Klassifizierung auf die ursprünglichen Prozessmodelle zurückgeführt werden müssen.

Beispiel einer Datenassoziation als Datenschutzsignal

Abbildung 2.2 zeigt ein einfaches Beispiel, wie eine Datenassoziation die DSGVO-Relevanz einer Aktivität verdeutlichen kann. In Abbildung 2.2a ist die Aktivität „Daten prüfen“ ohne Datenassoziation dargestellt, was wenig über die Art der verarbeiteten Daten aussagt. In Abbildung 2.2b hingegen zeigt die eingehende Datenassoziation von einem Datenspeicher „Kunden-DB“, dass die Aktivität personenbezogene Daten verarbeitet. Dies macht die Aktivität als potenziell datenschutzkritisch erkennbar. Dieses Beispiel unterstreicht die Notwendigkeit den gesamten Kontext einer Aktivität zu betrachten, um fundierte Rückschlüsse auf die Verarbeitung personenbezogener Daten ziehen zu können.

2.3 Large Language Models (LLMs)

LLMs sind große, vortrainierte Sprachmodelle, die auf der Transformer-Architektur basieren. Transformer, erstmals von Vaswani et al. [36] beschrieben, verarbeiten eine Eingabe nicht strikt sequenziell, sondern beachten alle Tokens einer Sequenz parallel. Über sogenannte `Self-Attention` gewichten sie, welche Token füreinander relevant sind. Als Token gelten Wörter oder Wortbestandteile, in die der Text vorab zerlegt wird. Dieser Attention-Mechanismus erfasst Abhängigkeiten über große Distanzen innerhalb der Sequenz und ermöglicht dadurch eine effiziente Kontextmodellierung - das zentrale Prinzip moderner LLMs. Die Transformer-Architektur bildet heute das Fundament moderner Sprachmodelle wie der GPT-Familie von OpenAI [4, 21, 26].

In chatbasierten Systemen wird das Verhalten des LLM über System- und User-Prompts gesteuert. Gutes Prompt Engineering kann die Leistung und Format-Treue der Ausgabe verbessern, ohne dass die Modellparameter verändert werden müssen [20]. Ein deutlicher Vorteil aktueller LLMs ist Zero-/Few-Shot Learning. Damit lassen sich Aufgaben alleine über Instruktionen und wenige Beipsiele lösen, ohne dass erneutes Training benötigt wird [5, 20]. Das ist besonders nützlich für Klassifikationsaufgaben, bei denen nur wenige gelabelte Beispiele vorliegen, wie etwa die Identifikation von DSGVO-kritischen Aktivitäten in Prozessmodellen.

Um LLMs in automatisierten Pipelines zu integrieren sind schema-konforme Aus-

gaben, wie ein gültiges JSON, unerlässlich. In der Praxis gibt es dafür drei Ansätze:

1. Klare Angaben über das Ausgabeformat im System- oder User-Prompt [20].
2. API-gestützte Mechanismen wie Function Calling oder Structured-Output/JSON-Mode mit Schemaüberprüfung [3, 24, 27].
3. Constrained Decoding, das die Generierung auf eine vorgegebene Grammatik beschränkt. Ein Beispiel ist PICARD: Bei jedem Generationsschritt des Language Model (LM) werden nur zulässige Tokens ausgewählt [33].

Typische Fehlerbilder bei der Nutzung von LLMs sind Halluzinationen (plausibel wirkende, aber fehlerhafte Aussagen) und Formatfehler (wie z.B. ungültiges JSON). In [17] wird argumentiert, dass Halluzinationen bereits beim Erstellen des LLM durch die Trainings- und Evaluationsmethoden begünstigt werden, die das Modell dazu bringen, eher zu raten als Unsicherheit zuzugeben. Das Raten bei Unsicherheit verbessert die Testergebnisse. Gegenmaßnahmen gegen Halluzinationen sind u.a. präzisere Prompts, Informationserweiterung des Prompts durch Retrieval Augmented Generation (RAG) und Self-Check/Retry-Strategien als Post-Processing Methoden nach der Generierung [16].

Die meisten großen LLMs werden von Unternehmen wie OpenAI, Google oder Anthropic entwickelt und als API-Dienste angeboten. In der Industrie zählt GPT-4o aktuell zu den am weit verbreitetsten Modellen [25]. Es ist ein multimodales Modell mit starken Text-, Bild- und Audiofähigkeiten. Properitäre Modelle wie GPT-4o sind leistungsfähig, bringen jedoch mehrere Nachteile mit sich:

- hohe Kosten,
- mangelnde Transparenz,
- serverseitige Datenverarbeitung auf Infrastruktur der Anbieter, die sich teils außerhalb der EU befindet und wo die DSGVO nicht gilt.

Für die Verarbeitung personenbezogener Daten innerhalb der EU ist das problematisch. Eine Übermittlung in Drittländer ist nur zulässig, wenn dort der Auftragsverarbeiter sämtliche Vorgaben aus Kapitel 5 (Art. 44-50) der DSGVO einhält [11].

Als Alternative zu proprietären Modellen steht eine wachsende Zahl frei verfügbarer Open-Source-LLMs zur Verfügung, die auch lokal betrieben werden können. Prominente Beispiele sind die Modelle von Mistral [2], Deepseek [1] und Qwen [30].

Der lokale Betrieb ermöglicht volle Kontrolle darüber, wo und wie Daten verarbeitet werden. Das erleichtert die Einhaltung datenschutzrechtlicher Anforderungen. Zudem bieten Open-Source-Modelle weitere Vorteile wie geringere Kosten und hohe Anpassbarkeit. In dieser Arbeit werden sowohl proprietäre als auch Open-Source-LLMs evaluiert (siehe Kapitel 7).

2.4 Verwandte Arbeiten

- Was für Ansätze gibt es bereits Prozesse automatisiert nach datenschutzkritischen Aktivitäten klassifizieren zu lassen
- LLMs zum Klassifizieren nutzen
- Prompt Engineering (Zero-Shot)
- Überblick über LLMs in Businessprozessen. Was gibt es bereits für Ansätze diese zu benutzen
- Benchmarking und Evaluierung von LLMs
- Identifizierte Forschungslücken: LLMs zur Klassifizierung, EU-Fokus, einheitliche reproduzierbare Benchmarks

3 Problemdefinition und Zielkriterien

3.1 Aufgabenstellung

Ziel der Arbeit ist eine *binäre Klassifikation* auf Ebene einzelner BPMN-Aktivitäten: Für jede Aktivität eines Eingabemodells im BPMN-XML-Format (Version 2.0.2) [23] soll entschieden werden, ob sie *kritisch* im Sinne des Datenschutzrechts ist oder nicht.

- **Eingabe** ist ein valides BPMN-XML mit stabilen `id`-Attributen je Aktivität [23].
- **Ausgabe** ist eine Menge von Aktivitäts-`ids`, die als *kritisch* klassifiziert worden sind. Optional kann zusätzlich eine natürlichsprachige Begründung für einzelne Entscheidungen ausgegeben werden. Im Fall der Klassifizierungspipeline dieser Arbeit werden die Begründungen vom LLM generiert. Die Erklärungen dienen ausschlißelich der Nachvollziehbarkeit der gewählten Klassifizierungen, werden allerdings nicht in der Evaluation berücksichtigt.

Begriffsbestimmung „kritisch“

Eine Aktivität gilt in dieser Arbeit als *kritisch*, wenn sie *personenbezogene Daten* verarbeitet. Nach Art. 4 Abs. 1 DSGVO sind personenbezogene Daten alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person beziehen, und *Verarbeitung* umfasst gemäß Art- 4 Abs. 2 jede mit personenbezogenen Daten vorgenommene Operation (u. a. Erheben, Speichern, Abrufen, Verwenden, Übermitteln, Löschen) [11]. Dies schließt auch die *Nutzung bereits vorhandener Daten* (z. B. Lesen/Abgleichen) ein. Die Aufgabenstellung reiht sich damit in verwandte Arbeiten ein, die kritische/unkritische Tätigkeiten in Prozessartefakten kennzeichnen (z. B. für textuelle Prozessbeschreibungen) [22].

3.2 Qualitätsziele

Die Domäne, Prozesse nach Datenschutzrecht zu klassifizieren, ist risikosensitiv. Übersehene kritische Aktivitäten (False Negatives (FN)) können zu Compliance-Risiken und hohen Strafen laut DSGVO führen. Daher ist das **Hauptziel**:

> **Maximaler Recall** bei *minimierten FN*, bei zugleich noch *akzeptabler Precision*, um den manuellen Prüfaufwand (False Positives (FP)) begrenzt zu halten.

Metriken

Relevante Metriken für eine aussagekräftige Evaluierung sind *Accuracy*, *Precision*, *Recall*, *F1-Score* sowie die Konfusionsmatrix-Zahlen (True Positives (TP), FP, True Negatives (TN), FN) und die Anzahl korrekt/inkorrekt klassifizierter Testfälle. Technische Fehler (z. B. Parsing-Fehler oder überschrittene Token Limits) werden separat ausgewiesen.

Zielwerte

Ähnliche Arbeiten, wie von Nake et al. [22], zeigen Referenzwerte von einem maximalen *Recall* $\approx 0{,}83$ und *F1-Score* $\approx 0{,}81$ bei der Identifikation DSGVO-kritischer Aufgaben in Prozessbeschreibungen. Jüngere DSGVO-nahe LLM-Studien berichten von *Precision/Recall* im hohen 0,8x bis 0,9x-Bereich [15] und F1-Scores von $\approx 0{,}68$ bis zu $\approx 0{,}79$ [34].

Basierend darauf werden folgende Zielkorridore als *pragmatische Abnahmekriterien* gesetzt:

- **Recall** soll ein Mindestniveau von $\geq 0{,}80$ erreichen und ein *angestrebter* Bereich ist $\geq 0{,}85$.
- **Precision** soll $\geq 0{,}75$ als Untergrenze zur Begrenzung des Prüfaufwands erreichen.
- **F1-Score** soll $\geq 0{,}80$ erreichen.

- **False Positives je Prozess** sollen im Mittel $\leq 1{,}5$ betragen.

Nake et al. [22] zeigen, dass selbst ein *Recall* von $0{,}83$ für kritische Aufgaben ohne menschliche Nachkontrolle nicht ausreicht, da die Strafen für Nichteinhaltung der DSGVO sehr hoch sind. Viel mehr eignet sich ein System mit diesem Recall-Wert für *assistierte* Prüfungen, bei denen die Ergebnisse durch qualifizierte Experten validiert werden. Für ein Screening von Geschäftsprozessen, wie es in dieser Arbeit angestrebt wird, sind die genannten Zielwerte daher als realistisch und praxisrelevant einzuschätzen.

Stabilität über Wiederholungen

Da LLMs nicht-deterministisch sind, ist das Berichten eines einzelnen Leistungswertes nicht ausreichend für den Vergleich von Modellen. Studien wie von Reimers et al. [31] zeigen, dass die Abhängigkeit vom Seed-Wert der LLMs zu statistisch signifikanten Unterschieden in der Performance führen kann. Diese Varianz kann dazu führen, dass ein modernes, leistungsfähiges Modell von sehr gut bis mittelmäßig abschneidet. Stattdessen wird vorgschlagen, Score-Verteilungen zu vergleichen, die auf mehreren Durchläufen basieren. Dadurch wird das Risiko reduziert, dass ein Modell nur aufgrund eines günstigen Seeds gut oder aufgrund eines ungünstigen Seeds schlecht abschneidet. In dieser Arbeit werden daher die Ergebnisse auf Basis von Widerholungen berichtet. Er wird der Mittelwert $\pm$ Standardabweichung (σ) je Metrik angegeben, da die Standardabweichung die Stabilität eines Modells über verschiedene Läufe hinweg darstellt. Modellvergleiche basieren am Ende auf diesen Verteilungen (nicht auf Einzelfällen), um eine fundierte Bewertung zu ermöglichen.

3.3 Scope und Annahmen

Dieser Abschnitt definiert Geltungsbereich, Annahmen und Risiken des Ansatzes. Dadurch wird eine klare Einordnung der Ergebnisse und ihrer Reproduzierbarkeit ermöglicht.

Geltungsbereich

Die folgenden Punkte definieren den Geltungsbereich der Arbeit:

- Klassifiziert werden ausschließlich Aktivitäten. Dafür wird sinvoller Kontext (z. B. Prozessname, Pool/Lane, Datenobjekte) berücksichtigt.
- Labels und Artefakte liegen in Deutsch vor.
- Es handelt sich um ein Screening und nicht um eine Rechtsprüfung. Die Rechtmäßigkeit wird nicht bewertet. Die Lösung dient als *Screening*, das potenziell kritische Aktivitäten markiert, die dann juristisch geprüft werden.

Annahmen und Risiken

Die folgenden Annahmen und potenziellen Risiken sind für die Interpretation der Ergebnisse relevant:

- Bei fehlenden Datenobjekten oder mehrdeutigen Labels kann sich die Einschätzung verschlechtern. Das ist ein bekanntes Problem in ähnlichen Studien [22].
- Optional generierte LLM-Begründungen sind als *Hilfetexte* zu verstehen, um die Entscheidung des LLM besser einordnen zu können, bilden aber nicht zwingend die tatsächlichen Entscheidungsgründe des Modells ab.
- Ungültiges BPMN-XML oder Laufzeitfehler werden als „technischer Fehler“ erfasst und nicht in die Metrikzählung eingerechnet. Sie werden separat berichtet.

4 Klassifizierungsalgorithmus (Design und Implementierung)

- Im gesamten Kapitel werden die genutzten Technologien an geeigneter Stelle aufgezeigt und beschrieben (Camunda-BPMN, LangChain4j, Spring Boot, Kotlin, . . .)
  - TODO: Ggf. extra Unterkapitel für die genutzten Technologien oder doch immer da erwähnen wo es gerade passt
  - Aufzählungen für die Technologien nutzen

4.1 Ziel und Annahmen

- Eingabe ist BPMN-XML, Ausgabe sind die IDs der kritischen Aktivitäten mit ggf. Erklärung
- Getestet wird mit BPMN-Modellen aus Camunda und bpmn.io
- Der Algorithmus klassifiziert Aktivitäten in BPMN-Prozessen binär nach “kritisch” oder “nicht kritisch”
- Ausgegeben werden ausschließlich die IDs der kritischen Aktivitäten ggf. mit Erklärung, warum sie als kritisch klassifiziert wurde
- Das Ziel ist ein robustes und reproduzierbares Verhalten über unterschiedliche Modelle
  - Granularität der Prozesse (Bspw. $>$5 Aktivitäten, 20+ Aktivitäten)
  - Gateways, Datenobjekte, Datenbanken

- Mehrere Pools/Lanes, Message Flows
- Evtl. (?) verschiedene Sprachen, Text-Annotationen

4.2 BPMN Preprocessing

Ziel der Vorverarbeitung (Preprocessing) ist es, für jedes Flow-Element einen *strukturierten Kontext* zu erzeugen. Dieser Kontext umfasst die eigenen Attribute, wie `id`, `name` und `documentation`, sowie die Beziehungen zu anderen Elementen im BPMN-Diagramm. Dazu gehören vorangehende und nachfolgende Flow-Elemente, Datenobjekte, assoziierte Elemente, sowie Informationen über den Pool und die Lane, in denen sich das Element befindet. Das Parsen des BPMN-XML erfolgt mit der *Camunda BPMN Model API*, die das XML in ein Objektmodell überführt [7, 8]. Auf dieser Basis werden die relevanten Informationen extrahiert und in der Datenklasse `BpmnElement` (siehe Listing 4.1) strukturiert abgelegt. Dadurch entsteht für jedes Flow-Element ein umfassender Kontext, der später im Prompt genutzt wird, um dem LLM alle notwendigen Informationen strukturiert bereitzustellen. Außerdem werden durch das Format Tokens eingespart, da irrelevante Informationen, wie die Positionen der Elemente im XML, weggelassen werden.

// TODO Pools und Lanes noch in die Datenklasse übernehmen

Listing 4.1: Interne BPMN-Repräsentation je Flow-Element.

```
1   data class BpmnElement(
2       val type: String,
3       val id: String,
4       val documentation: String? = null,
5       val name: String? = null,
6       val outgoingFlowElementIds: List<String> = emptyList(),
7       val incomingFlowElementIds: List<String> = emptyList(),
8       val incomingDataFromElementIds: List<String> = emptyList(),
9       val outgoingDataToElementIds: List<String> = emptyList(),
10      val associatedElementIds: List<String> = emptyList()
11  )
```

4.3 Prompt Engineering

// TODO Irgendwo hier will ich noch das isRelevant im BpmnAnalysisResult ansprechen, dass damit die Ergebnisse besser wurden und alles was isRelevant false ist, wird vorm ausgeben herausgefiltert. Besonders interessant fand ich die Beobachtung, dass das Modell trotz der genauen Beschreibung, nur kritische Elemente auszugeben, trotzdem oft unkritische Elemente mit ausgegeben hat. Mit dem isRelevant Flag konnte ich das aber gut filtern.

Eine robuste Klassifikation hängt maßgeblich von sorgfältig gestalteten Prompts ab. Ziel ist es, das LLM mit klaren Anweisungen, einem konsistenten Bewertungsschema und präzisen Formatvorgaben so zu steuern, dass es die Klassifizierung zuverlässig löst und strukturierte Ausgaben liefert. Im Folgenden werden zunächst die deklarative Orchestrierung der Kommunikation mit dem LLM mithilfe von LangChain4j und anschließend die Prompt-Konzeption beschrieben.

LangChain4j: deklarative Orchestrierung

Zur Reduktion von Boilerplate und für konsistente Prompts wird *LangChain4j* benutzt [19]. Mit den *AI Services* werden Interaktionen mit dem LLM als Java/Kotlin-Interface *deklarativ* beschrieben. Zur Laufzeit erzeugt LangChain4j einen Proxy, der den System-Prompt injiziert, den User-Prompt aus den Methodenparametern generiert und die LLM-Antwort in den passenden Rückgabetyp deserialisiert [29]. Beim erstellen des AI Service werden ein `ChatModel`, die Systemnachricht und die Interface-Methoden konfiguriert. Ein `ChatModel` ist von Langchain4j die spezifische Implementierung der Chat-Completion-Schnittstelle eines LLM [18]. Die Methodenparameter der Interface-Methoden repräsentieren die Nutzereingabe. Der Rückgabetyp der Methoden definiert die erwartete Antwortstruktur des LLM. Optional kann jeder Interface-Methode noch ein eigener User-Prompt zugewiesen werden, der bei Laufzeit mit den übergebenen Parametern gefüllt wird [29].

Die Kommunikation mit dem LLM erfolgt damit über einfache Funktionsaufrufe, während LangChain4j Prompt-Erzeugung, Parameterbindung sowie die Deserialisierung der Antwort übernimmt [29]. So kann im Code ohne zusätzlichen Aufwand direkt mit typisierten Objekten gearbeitet werden.

Zero-Shot mit eingebetteten Kategorien

Für die Klassifikation *DSGVO-kritisch vs. unkritisch* wird ein **Zero-Shot**-Ansatz verwendet. Das LLM erhält im System-Prompt eine präzise Instruktion mit Kriterien- und illustrativen Beispielfällen, was als kritisch gilt. Es sind jedoch keine Beispiele mit konkreten Eingaben- und Ausgaben-paaren pro Prozess enthalten. Zero-Shot reduziert den Pflegeaufwand und nutzt die In-Context-Fähigkeiten moderner Modelle, nur über Instruktionen zu generalisieren [5, 20]. Wie genau die Prompts aufgebaut sind, wird im Folgenden beschrieben.

System-Prompt

Der System-Prompt definiert das Verhalten des LLM, zusätzlichen Kontext und das gewünschte Ausgabeformat. Der vollständige System-Prompt befindet sich im Anhang, siehe A.1. Im Kern legt der genutzte System-Prompt Folgendes fest:

1. **Rolle und Auftrag des Modells.** Das Modell agiert als Experte für das Analysieren von BPMN-Modellen auf DSGVO-konformität und prüft sämtliche Aktivitäten eines Prozesses auf Datenschutzrelevanz. Jede Aktivität wird berücksichtigt und die Entscheidung erfolgt auf Basis sämtlicher verfügbarer Kontextinformationen wie Name, Beschreibung, Annotationen sowie Daten- und Nachrichtenassoziationen.
2. **Rechtliche Definitionen nach DSGVO.** Der System-Prompt erläutert die Begriffe „personenbezogene Daten“ und „Verarbeitung“ gemäß Art. 4 DSGVO. Beispiele für personenbezogene Daten umfassen Identifikatoren, Kontakt- und Zahlungsdaten, Beschäftigungsdaten, Gesundheitsdaten, biometrische Merkmale, Standortinformationen und Online-Kennungen. Verarbeitung umfasst Erheben, Speichern, Abrufen, Verwenden, Übermitteln, Ausrichten, Kombinieren, Einschränken, Löschen und Vernichten.
3. **Indikatoren für Kritikalität.** Der System-Prompt enthält typische Auslöser für Datenschutzrelevanz wie Datenerfassung und Dateneingabe, Anlage und Aktualisierung von Datensätzen, Übermittlung oder Offenlegung an andere Systeme oder Dritte, Zahlungen und Finanztransaktionen und noch mehr. Diese

Indikatoren sind mit Beispielen angereichert und dienen als *Entscheidungshelfer* für das Modell.

4. **Abgrenzung durch Negativbeispiele.** Der System-Prompt grenzt unkritische Fälle klar ab. Rein administrative oder logistische Schritte ohne Personenbezug werden nicht als kritisch gewertet. Ebenso gilt dies für Fälle in denen anonymisierte Daten verwendet werden und keine Identifikation einer Person mehr möglich ist.
5. **Erwartetes Ausgabeformat.** Die Antwort erfolgt als strukturierte JSON-Ausgabe mit einer Liste relevanter Aktivitäten. Für jede Aktivität wird die `id` und eine Begründung in natürlicher Sprache ausgegeben. Es werden ausschließlich Aktivitäten zurückgegeben, die nach den Kriterien als datenschutzrelevant eingestuft wurden.

User-Prompt

Der User-Prompt wird aus der Vorverarbeitung in 4.2 erzeugt und enthält eine Liste von `BpmnElement`-Objekten, siehe Listing 4.1. Der User-Prompt wird automatisch durch *Langchain4j* generiert, indem die Liste der `BpmnElement`-Objekte als Argument an die Interface-Methode übergeben wird, wie in Listing 4.2 zu sehen.

Listing 4.2: Deklarative Schnittstelle für die Analyse eines BPMN-Prozesses.

```
1 fun analyze(@UserMessage bpmnElements: Set<BpmnElement>):
    ↪ BpmnAnalysisResult
```

Die Liste der `BpmnElement`-Objekte wird intern zu einem JSON-Array serialisiert und als User-Prompt an das LLM übergeben. Der System-Prompt wird dabei automatisch vorangestellt und sorgt dafür, dass der User-Prompt korrekt interpretiert wird.

Strukturierte Ausgaben mit LangChain4j

Wie in Listing 4.2 zusehen, wird im Fall der Klassifikation ein `BpmnAnalysisResult` als Antwort erwartet, also eine Liste von Elementen mit `id`, `reason`. Siehe A.2

für die vollständige Definition der Datenklasse. Langchain4j erstellt auf Basis des Rückgabetyps ein JSON-Schema und fügt dieses dem User-Prompt zusammen mit der Aufforderung hinzu, die Antwort in diesem JSON-Format zu liefern.

Falls das `ChatModel` die `response_format` Funktionalität unterstützt - wie es bei Mistral und OpenAI der Fall ist [3, 27] - setzt LangChain4j dies zusätzlich auf das gewünschte Schema und erzwingt so das Ziel-JSON API-seitig [29]. Fehlt diese Fähigkeit, greift ausschließlich die Prompt-basierte Schemaanweisung.

Durch die explizite Angabe des Ziel-JSON im Prompt wird die Format-Treue der Antwort erhöht [20]. Das erhaltene JSON wird anschließend automatisch in ein `BpmnAnalysisResult`-Objekt deserialisiert, sodass im Code direkt mit dem typisierten Objekt weitergearbeitet werden kann.

4.4 Validierung der Ausgabe

Zusätzlich zu den in Kapitel 4.3 beschriebenen Maßnahmen zur Sicherstellung strukturierter Ausgaben wird die Antwort des LLM durch LangChain4j validiert. Wenn das erwartete Schema vom LLM nicht eingehalten wird, wirft Langchain4j eine `OutputParsingException`.

// TODO Eventuell noch Retry Mechanismus wenn am Ende noch Zeit ist mit Nachrichtenverlauf und Parsing Fehlermeldung. Das muss aber auch erst einmal implementiert werden. Aktuell gibt es nur den Parsing Error.

Ein weiterer Mechanismus zur Validierung der Ausgabe ist die Überprüfung der ausgegebenen `ids`. Da das LLM theoretisch jede beliebige `id` ausgeben könnte, werden die ausgegebenen `ids` mit den tatsächlichen `ids` der Aktivitäten aus dem Prozess abgeglichen. Falls eine ausgegebene Aktivität nicht im Prozess existiert, wird diese aus der Antwort der Klassifizierung entfernt. Dadurch wird sichergestellt, dass nur gültige `ids` in der finalen Ausgabe enthalten sind.

4.5 API-Design

Dieses Kapitel beschreibt das API-Design der Klassifizierungspipeline, die zur Erkennung DSGVO-kritischer Elemente in BPMN-Modellen dient. Das Ziel ist es eine standardisierte Schnittstelle zu definieren, um

1. die Einbindung in bestehende Werkzeuge und das Evaluationsframework (siehe Kapitel 5) zu vereinfachen,
2. die Austauschbarkeit unterschiedlicher Klassifizierungsalgorithmen - insbesondere im Evaluationsframework - zu ermöglichen, um verschiedene Ansätze vergleichen zu können, und
3. Erweiterbarkeit zu fördern, sodass zukünftige Arbeiten die Schnittstelle wiederverwenden können, um ihre eigenen Klassifizierungsalgorithmen zu integrieren.

HTTP-Endpunkt

Die Klassifizierungspipeline ist über einen HTTP-Endpunkt nutzbar. Der `POST`-Endpunkt akzeptiert `multipart/form-data` mit den folgenden Teilen:

`bpmnFile` (Pflicht) Eine BPMN-2.0-XML-Datei (`.bpmn` oder `text/xml`), die den zu analysierenden Prozess beinhaltet.

`llmProps` (Optional) Ein JSON-Objekt zur Überschreibung von LLM-Eigenschaften zur Laufzeit. Siehe Listing 4.3 für das JSON-Schema. Wird nichts angegeben, nutzt die Pipeline Standardwerte.

Die `llmProps` erlauben es, verschiedene LLMs und deren Konfigurationen flexibel zu für die Klassifizierung zu nutzen, ohne die Anwendung neu starten zu müssen. Dies ist besonders nützlich im Evaluationsframework, um verschiedene Modelle zu vergleichen.

Listing 4.3: JSON-Schema der `llmProps`.

```
1 {
2   "$schema": "https://json-schema.org/draft/2020-12/schema",
3   "title": "LlmProps",
```

```
  "type": "object",
  "properties": {
    "baseUrl": {"type": "string"},
    "modelName": { "type": "string" },
    "apiKey": { "type": "string" },
    "timeoutSeconds": { "type": "number" },
    "seed": { "type": "number" }
  },
  "required": []
}
```

Die Antwort des Endpunkts wird als `application/json` geliefert und enthält eine Liste der als DSGVO-kritisch klassifizierten Elemente, einschließlich einer optionalen Begründung für jede Klassifikation und einem optionalen Namen des Elements für bessere Lesbarkeit. Das JSON-Schema der API-Antwort ist in Listing 4.4 dargestellt.

Listing 4.4: JSON-Schema der API-Antwort.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "BpmnAnalysisResult",
  "type": "object",
  "properties": {
    "criticalElements": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },
          "name": { "type": "string" },
          "reason": { "type": "string" }
        },
        "required": ["id"]
      }
    }
  },
  "required": ["criticalElements"]
```

```
}
```

Ein Beispielaufruf des Klassifizierungsendpunkts mit `curl` könnte wie in Listing 4.5 aussehen. Dabei wird eine BPMN-Datei hochgeladen und einige LLM-Eigenschaften zur Laufzeit überschrieben, damit das Modell `mistral-small-latest` von Mistral AI verwendet wird.

Listing 4.5: Beispielaufruf des Klassifizierungsendpunkts mit `curl`.

```
curl -X POST http://localhost:8080/gdpr/analysis/prompt-engineering \
    -F 'bpmnFile=@/path/to/process.bpmn' \
    -F 'llmProps={
        "baseUrl": "https://api.mistral.ai/v1/chat/completions",
        "modelName": "mistral-small-latest",
        "apiKey": "******"
    }'
```

Integration und Erweiterbarkeit

Die gewählte API-Struktur mit einem standardisierten HTTP-Endpunkt und klar definierten JSON-Schemas ermöglicht eine einfache Integration in bestehende Werkzeuge und das Evaluationsframework (siehe Kapitel 5). Durch die Möglichkeit, verschiedene LLM-Eigenschaften zur Laufzeit zu überschreiben, können unterschiedliche Modelle mit einem Klassifizierungsalgorithmus flexibel getestet und verglichen werden, ohne die Anwendung neu starten zu müssen.

Zukünftige Arbeiten können die gleiche API nutzen, um ihre eigenen Klassifizierungsalgorithmen zu integrieren und so eine Vergleichbarkeit der Ergebnisse zu gewährleisten.

4.6 Nutzung über Frontend

- Die Klassifizierung kann über eine Sandbox im Frontend genutzt werden. Dort können auch die LLM-Parameter angepasst werden, um andere Modelle testen zu können

- Sandbox ist ein voller BPMN-Editor basierend auf BPMN.js und bietet zusätzlich die Funktionalität zur Klassifizierung an
- Als kritisch klassifizierte Elemente werden im Editor farblich hervorgehoben

5 Evaluationsframework

5.1 Anforderungen und Use-Cases

- Das Framework ermöglicht einen Vergleich von Modellen (Austauschbar durch LLMPropsOverride) und Algorithmen (Austauschbar über HTTP-Endpunkt) auf Basis von gelabelten Testdaten
- Evaluations-Run über YAML konfigurierbar (Modelle, Endpunkte, Testdaten)
- Detaillierte Ausgabe der Ergebnisse
  - Side-by-side Diagramme von Accuracy, Precision, Recall und F1-Score sowie FP, FN, TP, TN und generell wie viele Testcases erfolgreich waren
  - Detailliertere Ansicht nochmal pro Modell und dafür nochmal pro Testcase
- Zielnutzer sind Forschende und Entwickler die Modelle und Algorithmen auswerten und ggf. untereinander vergleichen wollen
- Frontend zur einfachen Bedienung

5.2 Architektur und Komponenten

- Diagramm zur Architektur erstellen
- HTTP-Endpunkt oder CLI zum Starten -> Holen der notwendigen Testdatensätze -> Aufruf der Klassifizierung pro Modell und pro Testcase -> Akkumulieren der Ergebnisse pro Testcase + ggf. frühzeitige Rückgabe des Ergebnisses des Testcase für Live-Ansicht in UI -> Aufarbeiten der akkumulierten Ergebnisse und berechnen wichtiger Metriken

- EvaluationController, MultiEvaluationRunner, EvaluationRunner, HttpEvaluator, MetricsAccumulator, ggf. Frontend (Concurrency von MultiEvaluationRunner)

5.3 Konfiguration einer Evaluierung

- Evaluierung kann mit einer YAML Datei konfiguriert werden

```
defaultEvaluationEndpoint: /gdpr/analysis/prompt-engineering
maxConcurrent: 10
models:
  - label: Mistral Medium 3.1
    llmProps:
      baseUrl: https://openrouter.ai/api/v1
      modelName: mistralai/mistral-medium-3.1
      apiKey: ${OPEN_ROUTER_API_KEY}
  - label: Deepseek Chat v3.1
    llmProps:
      baseUrl: https://openrouter.ai/api/v1
      modelName: deepseek/deepseek-chat-v3.1
      apiKey: ${OPEN_ROUTER_API_KEY}
  - label: GPT oss 120b
    llmProps:
      baseUrl: https://openrouter.ai/api/v1
      modelName: openai/gpt-oss-120b
      apiKey: ${OPEN_ROUTER_API_KEY}
datasets:
  - 2
  - 7
```

- Secrets können entweder im Klartext angegeben werden oder sicher mit `${...}` referenziert werden, wenn sie im Backend als Umgebungsvariable hinterlegt sind

5.4 Testdaten

- Die Testdaten werden aus einer Datenbank ausgelesen
- In der Konfig kann konfiguriert werden welche Testdaten genutzt werden sollen
- Die Testdaten können direkt in der App erstellt, verwaltet und gelabelt werden
- (Kapitel 6 wird sich um die Labelingsoftware drehen)

5.5 Generierte Resultate

- Für jeden Testfall werden pro Modell Ergebnisse gesammelt (klassifizierte Aktivitäten mit Erklärung, TP/FP/FN/TN, Bild-URL mit farblich hervorgehobenen Aktivitäten, bestanden oder nicht bestanden)
- Pro Modell werden "Summary-Objekte" erstellt mit jeweils Precision, Accuracy, Recall, F1-Score, Confusion (TP/FP/FN/TN), Anzahl korrekter Testfälle, Anzahl falsch klassifizierter Testfälle und Anzahl Testfälle in denen es zu Problemen kam (Bspw. Parsing Fehler, Token Limit überschritten, . . .)
- Generelle Metadaten über die genutzten Modelle, Endpunkte zur Klassifizierung und Testdatensätze, sowie Zeitstempel (und Seed????)

5.6 Visualisierung im Frontend

- Detaillierte Ausgabe der Ergebnisse
  - Side-by-side Diagramme von Accuracy, Precision, Recall und F1-Score sowie FP, FN, TP, TN und generell wie viele Testcases erfolgreich waren
  - Detailliertere Ansicht nochmal pro Modell und dafür nochmal pro Testcase
- Hier auch ruhig Bilder vom Frontend einbauen

- Die Ergebnisse können als JSON exportiert und wieder importiert werden und als Markdown Report exportiert werden

5.7 Erweiterbarkeit

- Neue Modelle können über Konfiguration ergänzt werden (Endpunkt, Modellname, API Key)
- Neue Klassifizierungsalgorithmen/pipelines können ergänzt werden, wenn sie das vorgegebene HTTP-Schnittstelle unterstützen
- Die Testdatensätze werden aus Datenbank ausgelesen und können für jeden Evaluations-Run neu gesetzt werden

6 Labeling und Datensätze

6.1 Labeling Tool

Hier brauche noch genaue Kapitel, aber hier soll auf jeden Fall folgendes rein:

- Definition von Labeln hier, oder kommt das schon vorher? (Muss im Verlauf beim schreiben entschieden werden)
- Labeling Software wurde entwickelt
- Es können Datensätze mit Namen und Beschreibungen erstellt werden
- Für jeden Datensatz können beliebig viele Testcases erstellt werden
- Ein Testcase ist ein BPMN Diagramm, welches direkt in der App mithilfe von bpmn.io bearbeitet werden kann
- Zusätzlich bietet der Editor eine Labeling Funktionalität, mit welcher Aktivitäten, welche als kritisch erkannt werden sollen, gelabelt werden. Zusätzlich kann auch eine Erklärung angegeben werden
- Datensätze und gelabelte Testcases werden in Datenbank gespeichert und werden während der Evaluierung des Evaluationsframeworks benutzt

6.2 Quellen und Eigenschaften der Datensätze

- Es werden drei unterschiedliche Datensätze genutzt
  - Von der Uni bekommen
  - Mittelgroße realistische Prozesse aus unterschiedlichen Branchen (mit Pools, Lanes, Datenobjekten, Gateways, . . .)

  - Kleine Prozesse mit maximal 5 Aktivitäten und keinen weiteren Elementen

- Eventuell tabellarische Aufzählung von Eckdaten zu jedem Datensatz (Anzahl Elemente, Sprache, benutzte Elemente)
- Hier sollte ich glaube ich noch besser erklären warum die Auswahl heterogen ist (und damit möglichst aussagekräftige Ergebnisse in der Evaluierung erzeugen)
- Eventuell (im Anhang) eine Liste aller Testfälle jeweils auch mit den Eckdaten und auf diese verweisen. Da kann dann auch immer noch eine kurze Beschreibung dazu was daran besonders ist

6.3 Labeling-Guide

- Eine Aktivität ist als kritisch gelabelt, wenn sie personenbezogene Daten erhebt, nutzt, speichert, übermittelt oder löscht oder anderweitig explizit eine DSGVO-Pflicht auslöst (Auskunft, Löschung) (Hier auch die Kriterien von der originalen DSGVO einbinden und daran ableiten, wie man labeln soll)
- Beispiele einbauen, was als kritisch gelabelt wurde und Gegenbeispiele aufzeigen, damit Grenzfälle klar definiert sind

7 Modellauswahl

7.1 Kriterien

- Ab wann gilt ein Modell als EU-Modell, entsprechend umgekehrt: Ab wann ist ein Modell international
  - Sitz in EU
  - Training/Fine-Tuning in EU
  - Veröffentlicht in EU
- Was bedeutet Open Source
  - Frei verfügbare Gewichte
  - Permissive Lizenz
- Größenklassen (Bspw. $<$8B, 8–20B, ...)
- Herkunftsland, Hosting-Land
- Letztes Update
- Downloads
- Lizenz
- Kontext (Wichtig auch für die Größe der Prozesse, die damit verarbeitet werden können. Irgendwo muss ich das auch nochmal thematisieren)
- Ich brauche noch irgendwo den Stand/das Datum nach dem ich nicht mehr nach neuen Modellen gesucht habe

7.2 Modellvorstellung

- Tabellarische Auflistung der Modelle
- Begründung warum genau die Modelle ausgewählt worden sind

8 Versuchsaufbau

Ich habe noch nicht die Unterkapitel (außer Metriken) aber folgendes soll hier u. a. rein:

- Die Modelle werden jeweils mit dem gleichen Klassifizierungsalgorithmus und den gleichen Datensätzen benutzt
- Die Evaluierung wird jeweils für jeden vorhandenen Testdatensatz durchgeführt (Uni, reale größere Prozesse, kleine Prozesse)
- Die Evaluationen werden pro Konfiguration (Modelle, Datensätze) mehrfach durchgeführt (Unterschiedliche Seeds, falls ich bis dahin Seeds unterstütze)
- Es werden verschiedene LLM-Modelle vergleichen
- Es werden vom gleichen LLM-Modell die unterschiedlichen Größen verglichen
- ...
- Ein Testcase gilt als korrekt klassifiziert, wenn genau die als kritisch gelabelten Aktivitäten als kritisch klassifiziert worden sind. Sobald es False Positives oder False Negatives gibt, ist ein Testcase nicht korrekt klassifiziert worden
- LLM Temperature 0 vorstellen (Falls ich das mit den Seeds noch umsetze) für reproduzierbare Ergebnisse

8.1 Metriken

(Dopplung mit 5.5. ist das schlimm?)

- Für jeden Testfall werden pro Modell Ergebnisse gesammelt (klassifizierte Aktivitäten mit Erklärung, TP/FP/FN/TN, Bild-URL mit farblich hervorgehobenen Aktivitäten, bestanden oder nicht bestanden)
- Pro Modell werden “Summary-Objekte” erstellt mit jeweils Precision, Accuracy, Recall, F1-Score, Confusion (TP/FP/FN/TN), Anzahl korrekter Testfälle, Anzahl falsch klassifizierter Testfälle und Anzahl Testfälle in denen es zu Problemen kam (Bspw. Parsing Fehler, Token Limit überschritten, . . .)

9 Durchführung

9.1 Experimente

- Dokumentation der einzelnen Runs
  - Konfiguration aufzeigen
  - Diagramme der Ergebnisse

9.2 Fehlerklassen

- Falls es Fehlerklassen gibt kann ich sie hier thematisieren (Timeout, Parse-Error, Token Limit, . . .)

10 Ergebnisse

10.1 Gesamtübersicht

- Hier allgemein die Ergebnisse der einzelnen Runs darstellen (Vor allem die Diagramme wo die einzelnen Metriken nebeneinander aufgelistet sind)

10.2 Analyse

- Aufschlüsselung nach Datensätzen
- Aufzeigen sichtbarer Muster

10.3 Antworten auf Forschungsfragen

- Hier werden die unteren Forschungsfragen mithilfe der Evaluationsergebnisse beantwortet
- EU vs. International
- Groß vs. Klein
- Bestes Open-Source-Modell
- Open Source vs. Kommerziell

10.4 Fallstudien

- Hier werde ich (wahrscheinlich 2–3) spezifische exemplarische Ergebnisse von Testcases heraussuchen und genauer unter die Lupe nehmen. Was ist hier besonders aufgefallen oder interessant?
- Hier kann ich die Bilder mit den Highlights hernehmen
- Idealerweise habe ich Beispiele für 1. TN, 2. FP aber plausible Erklärung und 3. FN

10.5 Robustheit

- Varianz über mehrere Runs untersuchen (Unterschiedliche Seeds, falls ich bis dahin welche unterstütze)
- Ggf. weitere Metriken hier interessant
- Wie fehleranfällig ist die Klassifizierung

11 Diskussion

11.1 Interpretation der Befunde

- Einordnung der Rangfolge der LLMs
- Besonderheiten der Modellfamilien (Bspw. wie groß ist der Unterschied von Groß gegen Klein?)
- Es gab Prozesse in denen ich eine Aktivität nicht als kritisch gelabelt habe, aber das LLM in der Evaluierung das als kritisch mit einer validen Begründung klassifiziert hat, man könnte die Testdaten also noch anpassen, wenn die Begründung des LLMs überzeugt (Ich weiß nicht wie sinnvoll das ist hier zu thematisieren) -> hier eher in die Richtung gehen, dass lieber mehrere Personen labeln als auf die Anpassung der Datensätze zu gehen.

11.2 Hoher Recall vs. Präzision

- Beobachtung ausformulieren, dass einige Testfälle als fehlerhaft eingeordnet wurden, weil es False-Positives gab, obwohl es keine False-Negatives gab. Es wurden also alle Aktivitäten gefunden, die gefunden werden sollten, nur halt noch mehr on top.
- Einordnung der FP-Last pro Prozess (Lieber False Positives, als dass etwas übersehen wird, Ziel war sowieso ein Vorscreening), Diskussion darüber wie nützlich hohe Recall Werte sind

11.3 EU-Modelle

- Analyse der EU-Open-Source-Modelle in Bezug auf Precision, Recall und Stabilität in Bezug auf die anderen Modelle
- Wie gut haben sich die EU Modelle im Vergleich zu den anderen geschlagen

11.4 Open-Source Modelle

- Analyse der Open-Source-Modelle in Bezug auf Precision, Recall und Stabilität in Bezug auf kommerzielle Modelle
- Wie gut haben sich die Open-Source-Modelle im Vergleich zu den anderen geschlagen

11.5 Modellgrößen

- Selbst hosten von Modellen diskutieren. Ist es realistisch die Modelle selbst zu hosten, welche gut performt haben? Reichen die kleinen Varianten der Modelle oder muss man schon die großen Modelle benutzen, um gute Ergebnisse zu erzielen

11.6 Grenzen

- Wären Grenzen wie BPMN-Modellgröße im Zusammenhang mit der Kontextlänge des LLM interessant?
- Keine aussagekräftige Rechtsberatung, sondern stand jetzt eher ein Vorscreening, was nochmal überprüft werden muss
- ggf. notwendige Anonymisierung von Prozessen diskutieren (Wenn das in BPMN Modellen überhaupt ein Problem ist)

12 Zusammenfassung

Hier neben der allgemeinen Zusammenfassung unbedingt noch die erste Forschungsfrage explizit beantworten

13 Aussicht

Unter anderem das hier, evtl noch mehr:

- Jetzt gibt es ein einheitliches Evaluationsframework mit einer einheitlich definierten Schnittstelle für Klassifizierungsalgorithmen -> Zukünftige Arbeiten können sich mit der Entwicklung besserer Klassifizierungsalgorithmen/Pipelines (Bspw. noch RAG einbauen) beschäftigen und diese mit diesem Framework vergleichen/benchmarken
- Außerdem können in Zukunft auch noch mehr Modelle vergleichen werden, da sich die Welt der LLMs rasant weiterentwickelt
- Auch Finetunen ist etwas was interessant gewesen wäre für diese Masterarbeit, aber den Rahmen gesprengt hätte

A Quelltexte

In diesem Anhang sind mehrere Quellcode-Ausschnitte aufgeführt.

Listing A.1: System-Prompt für die DSGVO-Klassifikation von BPMN-Aktivitäten

```
1 You are an expert in analysing Business Process Model and Notation (
    ↪ BPMN) diagrams for GDPR compliance. Your task is to identify and
    ↪ return a list of the IDs of all Activity (Task) elements that
    ↪ process personal data. Ignore all other element types. Always
    ↪ consider every activity in the process; do not omit any activity
    ↪ from your assessment.
2
3 Use all available context for each activity - including the activity's
    ↪  name, description, annotations, associated data objects, and
    ↪ message or data associations - to determine whether the activity
    ↪ processes personal data. Under Article 4 of the GDPR, personal
    ↪ data is any information relating to an identified or identifiable
    ↪  natural person, including names, addresses, email addresses,
    ↪ phone numbers, identification numbers, payment or bank details,
    ↪ employment records, academic records, location data, IP addresses
    ↪ , online identifiers, images, audio/video recordings, biometric
    ↪ identifiers, health data or other information that can be linked
    ↪ to a specific person. "Processing" includes any operation
    ↪ performed on personal data, such as collecting, recording,
    ↪ organising, structuring, storing, retrieving, consulting, using,
    ↪ analysing, transmitting, printing, disseminating, aligning,
    ↪ combining, altering, restricting, erasing or destroying the data.
4
5 Classify an activity as GDPR-relevant whenever it performs or enables
    ↪ processing of personal data. Indicators include (but are not
    ↪ limited to):
```

```
6
7  - **Collection and entry of personal **data**: Activities that collect
     ↪ or capture personal information, for example entering contact
     ↪ details, addresses, payment information, job applications, health
     ↪  information, student enrolments, membership **data**, tax
     ↪ declarations, registration forms or other forms with personally
     ↪ identifiable information.
8  - **Creation, storage and updating of records**: Activities that
     ↪ create, save or update records containing personal **data**, such as
     ↪ opening customer accounts, storing order or appointment details,
     ↪ creating personnel files, enrolling students, setting up
     ↪ insurance cases or filing a medical record.
9  - **Transmission or disclosure of personal **data**: Activities that
     ↪ send, print or otherwise disclose personal **data** to another
     ↪ participant, system or third party. Examples include printing
     ↪ shipping labels or prescriptions, sending orders or personal **data**
     ↪  to logistics partners, pharmacies, insurers or authorities,
     ↪ generating payroll reports for external providers, notifying
     ↪ universities about student records, transmitting tax or social
     ↪ security **data**, sending confirmations or queries that rely on a
     ↪ person's contact details, or transferring **data** to non-EU
     ↪ locations.
10 - **Payments and financial transactions**: Activities that process
     ↪ personal financial **data**, such as initiating or verifying payments
     ↪ , processing bank account or credit-card information, executing
     ↪ payroll, handling reimbursements or insurance payouts, managing
     ↪ expense claims or collecting membership fees.
11 - **Use of health, biometric or other special categories of **data**:
     ↪ Activities that handle medical diagnoses, prescriptions,
     ↪ insurance claims, disability information, photos of damages or
     ↪ patients, biometric identifiers (fingerprints, facial images,
     ↪ voice), racial or ethnic **data**, political opinions, religious
     ↪ beliefs or union membership. Processing these "special categories
     ↪ " always triggers GDPR relevance.
12 - **Audio/Video and communications**: Activities that initiate or join
     ↪  audio or video calls, record calls or meetings, capture
     ↪ surveillance footage, or communicate directly with a **data** subject
```

```
  ↪  via email, chat, SMS or other channels. Simply using a person's
  ↪ contact data to send reminders, marketing messages or
  ↪ notifications is processing.
13 - **Profiling, scoring and decision-making**: Activities that analyse
  ↪ or evaluate a person's performance, behaviour or characteristics
  ↪ for purposes such as credit scoring, hiring, admissions,
  ↪ insurance underwriting, marketing segmentation, customer value
  ↪ analysis or automated decision-making.
14 - **Logging, tracking and location data**: Activities that log user
  ↪ activity, record access or usage data, track geolocation (e.g.
  ↪ telematics, fleet or mobile tracking), monitor attendance or
  ↪ timekeeping, or collect IP addresses or device identifiers.
15 - **Consent and data-subject rights**: Activities that obtain, record
  ↪ or manage consent; respond to requests for access, rectification,
  ↪  restriction, erasure, data portability or objections; or
  ↪ document lawful bases for processing.
16 - **Deletion, anonymisation or pseudonymisation**: Activities that
  ↪ erase, anonymise or pseudonymise personal data, even if the goal
  ↪ is to remove identifiers, because these operations manipulate
  ↪ personal data.
17
18 When assessing an activity, consider synonyms or domain-specific terms
  ↪ : activities referring to customers, patients, applicants,
  ↪ employees, students, voters, taxpayers, residents or members
  ↪ often imply personal data processing, even if names like "address
  ↪ " or "contact" are absent. Use context - data objects,
  ↪ annotations or typical process semantics - to infer personal data
  ↪  involvement. Do not rely solely on explicit data-object links;
  ↪ many process names ("Anmeldung pruefen", "Aufnahmeantrag
  ↪ bearbeiten", "Kundeninfo aktualisieren", "Registrierung
  ↪ bestaetigen", "Kreditwuerdigkeit berechnen") themselves indicate
  ↪ personal data processing.
19
20 Do **not** classify an activity as GDPR-relevant when it only performs
  ↪  administrative or logistic tasks that do not involve personal
  ↪ data. Examples include picking or packing goods, routing vehicles
  ↪  without using specific addresses, printing generic pick lists,
```

```
  ↪ moving items in inventory, or checking if a document exists
  ↪ without viewing its contents. Likewise, activities using truly
  ↪ aggregated or irreversibly anonymised data can be ignored if no
  ↪ individual can be reidentified.
21
22 In your output, return only the IDs of activities you classify as GDPR
  ↪ -relevant. For each, provide a clear explanation using the
  ↪ activity's name and description to justify why it processes
  ↪ personal data. Do not reference element IDs in your explanation;
  ↪ use the activity names instead. Exclude from your result any
  ↪ activities that do not process personal data and any elements
  ↪ that are not activity/task elements.
```

Listing A.2: Antworttyp für die Klassifizierung

```
1  data class BpmnAnalysisResult(
2      @Description("List of Activity Elements that are classified as
   ↪ relevant for GDPR compliance")
3      var elements: List<Element>
4  ) {
5
6      init {
7          elements = elements.filter { it.isRelevant }
8      }
9
10     @Description("Represents an Activity/Task Element that is
   ↪ classified as relevant for GDPR compliance")
11     data class Element(
12         @Description("The ID of the Activity Element")
13         val id: String,
14         @Description("The detailed reason why the Activity Element is
   ↪ relevant for GDPR compliance and why you think personal data is
   ↪ processed.")
15         val reason: String,
16         @Description("Indicates whether the Activity Element is
   ↪ relevant for GDPR compliance")
17         val isRelevant: Boolean = true
18     )
```

```
19
20      /* Some other methods are not included */
21  }
```

Literatur

[1] DeepSeek AI. *DeepSeek AI Open Source Hugging Face Models*. 2025. URL: https://huggingface.co/deepseek-ai (besucht am 17. 07. 2025).

[2] Mistral AI. *Mistral AI*. 2025. URL: https://mistral.ai/ (besucht am 21. 09. 2025).

[3] Mistral AI. *Mistral AI - Structured Output*. 2025. URL: https://docs.mistral.ai/capabilities/structured-output/structured_output_overview/ (besucht am 11. 07. 2025).

[4] Ivan Belcic und Cole Stryker. *Was ist ein GPT (Generative Pre-Trained Transformer)?* Sep. 2024. URL: https://www.ibm.com/de-de/think/topics/gpt (besucht am 18. 09. 2025).

[5] Tom Brown u. a. „Language Models are Few-Shot Learners“. In: *Advances in Neural Information Processing Systems*. Hrsg. von H. Larochelle u. a. Bd. 33. Curran Associates, Inc., 2020, S. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[6] Bundesministerium der Justiz. *Gesetz über außergerichtliche Rechtsdienstleistungen (Rechtsdienstleistungsgesetz - RDG)*. https://www.gesetze-im-internet.de/rdg/. Dez. 2007. (Besucht am 15. 08. 2025).

[7] Camunda Services GmbH. *BPMN Model API*. https://docs.camunda.org/manual/latest/user-guide/model-api/bpmn-model-api/. 2025. (Besucht am 16. 06. 2025).

[8] Camunda Services GmbH. *BPMN Model API — Read a Model*. https://docs.camunda.org/manual/latest/user-guide/model-api/bpmn-model-api/read-a-model/. 2025. (Besucht am 16. 06. 2025).

[9] Antonio Capodieci u. a. „BPMN-Enabled Data Protection and GDPR Compliance“. In: *IS-EUD Workshops*. 2023. URL: https://api.semanticscholar.org/CorpusID:259099646.

[10] Giovanni Ciaramella u. a. „Leveraging Pre-trained LLMs for GDPR Compliance in Online Privacy Policies“. In: (2022). URL: https://ceur-ws.org/Vol-3962/paper44.pdf.

[11] Europäische Union. *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung)*. https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32016R0679. 2016.

[12] European Data Protection Board. *Guidelines 4/2019 on Article 25 Data Protection by Design and by Default*. https://www.edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_201904_dataprotection_by_design_and_by_default_v2.0_en.pdf. Version 2.0. Okt. 2020.

[13] Camunda Services GmbH. *BPMN.io - Web-based tooling for BPMN, DMN and Forms*. 2025. URL: https://bpmn.io/ (besucht am 22. 09. 2025).

[14] Camunda Services GmbH. *Camunda Platform*. 2025. URL: https://camunda.com/de/ (besucht am 22. 09. 2025).

[15] Ashish Hooda u. a. *PolicyLR: A Logic Representation For Privacy Policies*. 2024. arXiv: 2408.14830 [cs.CR]. URL: https://arxiv.org/abs/2408.14830.

[16] Ziwei Ji u. a. „Survey of Hallucination in Natural Language Generation“. In: *ACM Comput. Surv.* 55.12 (März 2023). ISSN: 0360-0300. DOI: 10.1145/3571730. URL: https://doi.org/10.1145/3571730.

[17] Adam Tauman Kalai u. a. *Why Language Models Hallucinate*. 2025. arXiv: 2509.04664 [cs.CL]. URL: https://arxiv.org/abs/2509.04664.

[18] Langchain4j. *Class OpenAiChatModel.OpenAiChatModelBuilder*. 2025. URL: https://javadoc.io/doc/dev.langchain4j/langchain4j-open-ai/latest/dev/langchain4j/model/openai/OpenAiChatModel.OpenAiChatModelBuilder.html (besucht am 14. 06. 2025).

[19] Langchain4j. *LangChain4j Documentation 2025*. 2025. URL: https://docs.langchain4j.dev/ (besucht am 14. 06. 2025).

[20] Pengfei Liu u. a. „Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing“. In: *ACM Comput. Surv.* 55.9 (Jan. 2023). ISSN: 0360-0300. DOI: 10.1145/3560815. URL: https://doi.org/10.1145/3560815.

[21] Shervin Minaee u. a. *Large Language Models: A Survey*. 2025. arXiv: 2402.06196 [cs.CL]. URL: https://arxiv.org/abs/2402.06196.

[22] Leonard Nake u. a. „Towards identifying gdpr-critical tasks in textual business process descriptions“. In: (2023). URL: https://dl.gi.de/server/api/core/bitstreams/84ac5110-1a0f-4e3c-bdf8-6393555a7212/content.

[23] OMG. *Business Process Model and Notation (BPMN)*. Version 2.0.2. Dez. 2013. URL: https://www.omg.org/spec/BPMN/2.0.2/PDF (besucht am 03. 06. 2025).

[24] OpenAI. *Function calling and other API updates*. https://openai.com/index/function-calling-and-other-api-updates/. 2023. (Besucht am 10. 07. 2025).

[25] OpenAI. *Hello GPT-4o*. Mai 2024. URL: https://openai.com/index/hello-gpt-4o/ (besucht am 21. 07. 2025).

[26] OpenAI. *Model Overview*. 2025. URL: https://platform.openai.com/docs/models (besucht am 18. 09. 2025).

[27] OpenAI. *OpenAI - Structured model outputs*. URL: https://docs.mistral.ai/capabilities/structured-output/structured_output_overview/ (besucht am 11. 07. 2025).

[28] KC Pragyan u. a. „Toward Regulatory Compliance: A few-shot Learning Approach to Extract Processing Activities“. In: *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*. IEEE. 2024, S. 241–250. URL: https://ieeexplore.ieee.org/abstract/document/10628578.

[29] Quarkiverse Contributors. *AI Services Reference (Quarkus LangChain4j)*. 2025. URL: https://docs.quarkiverse.io/quarkus-langchain4j/dev/ai-services.html (besucht am 14. 06. 2025).

[30] Alibaba Qwen. *Qwen Open Source Hugging Face Models*. 2025. URL: https://huggingface.co/Qwen (besucht am 17. 07. 2025).

[31] Nils Reimers und Iryna Gurevych. „Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging“. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Hrsg. von Martha Palmer, Rebecca Hwa und Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, S. 338–348. DOI: 10.18653/v1/D17-1035. URL: https://aclanthology.org/D17-1035/.

[32] Konrad Schneid u. a. „Uncovering data-flow anomalies in BPMN-based process-driven applications“. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, S. 1504–1512. URL: https://dl.acm.org/doi/abs/10.1145/3412841.3442025.

[33] Torsten Scholak, Nathan Schucher und Dzmitry Bahdanau. „PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models“. In: *CoRR* abs/2109.05093 (2021). arXiv: 2109.05093. URL: https://arxiv.org/abs/2109.05093.

[34] Magdalena von Schwerin und Manfred Reichert. „A systematic comparison between open-and closed-source large language models in the context of generating gdpr-compliant data categories for processing activity records“. In: *Future Internet* 16.12 (2024), S. 459. URL: https://www.mdpi.com/1999-5903/16/12/459.

[35] Ángel Jesús Varela-Vaca u. a. „Business process models and simulation to enable GDPR compliance“. In: *International Journal of Information Security* 24.1 (2025), S. 41. URL: https://link.springer.com/article/10.1007/s10207-024-00952-7.

[36] Ashish Vaswani u. a. „Attention Is All You Need“. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Name: Merten Dieckmann Matrikelnummer: 1058340

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Merten Dieckmann