



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 5. Oktober 2025

© 2025 Merten Dieckmann

Satz: PDF- \LaTeX 2 _{ε}

Inhaltsverzeichnis

Abkürzungen	vi
1 Einleitung	2
1.1 Motivation	2
1.2 Problemstellung	3
1.3 Zielsetzung und Beiträge	4
1.4 Forschungsfrage und Unterfragen	4
1.5 Aufbau der Arbeit	5
2 Hintergrund und verwandte Arbeiten	6
2.1 Datenschutzgrundverordnung (DSGVO)	6
2.2 Business Process Model and Notation (BPMN)	7
2.3 Large Language Models (LLMs)	10
2.4 Verwandte Arbeiten	12
3 Problemdefinition und Zielkriterien	13
3.1 Aufgabenstellung	13
3.2 Qualitätsziele	14
3.3 Scope und Annahmen	17
3.4 Experimentdesign	18
4 Design und Implementierung der Klassifizierungspipeline	20
4.1 BPMN Preprocessing	20
4.2 Prompt Engineering	22
4.3 Validierung der Ausgabe	26
4.4 API-Design	29
4.5 Nutzung über Webapp	31

5	Labeling und Datensätze	34
5.1	Labeling Tool	34
5.2	Quellen und Eigenschaften der Datensätze	35
5.3	Labeling-Guide	37
6	Evaluationsframework	39
6.1	Use-Cases und Anforderungen	39
6.2	Testdaten	42
6.3	Konfiguration einer Evaluierung	42
6.4	Architektur und Komponenten	44
6.5	Evaluationsergebnisse	47
6.6	Frontend	49
6.7	Erweiterbarkeit	54
7	Modellauswahl	55
7.1	Kriterien	55
7.2	Modellvorstellung	56
8	Versuchsaufbau	57
9	Durchführung	58
9.1	Experimente	58
9.2	Fehlerklassen	58
10	Ergebnisse	59
10.1	Gesamtübersicht	59
10.2	Analyse	59
10.3	Antworten auf Forschungsfragen	59
10.4	Fallstudien	60
10.5	Robustheit	60
11	Diskussion	61
11.1	Interpretation der Befunde	61
11.2	Hoher Recall vs. Präzision	61
11.3	EU-Modelle	62
11.4	Open-Source Modelle	62
11.5	Modellgrößen	62

Inhaltsverzeichnis

11.6 Grenzen	62
12 Zusammenfassung	63
13 Aussicht	64
A Quelltexte	65
Literatur	76

Abkürzungen

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

Abkürzungen

// TODO Überall nach einheitlichen Begriffsnutzungen schauen (Modell, Prozessmodell, LLMs. BPMN-Prozess, BPMN-Modell, Labels, Annotationen, Evaluation, Experiment)

// TODO Sprache in Bildern/der App und Diagrammen einheitlich anpassen. Entweder es soll alles auf Englisch sein oder auf Deutsch genau wie der Text.

1 Einleitung

Dieses Kapitel leitet thematisch in die Arbeit ein und gibt einen Überblick über die Ziele und die Struktur.

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 [11, 14]. 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 [14].

Die Überprüfung von Prozessen auf Konformität in Bezug auf Datenschutz ist jedoch zeit- und kostenintensiv [27, 43]. 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 [27].

Vor diesem Hintergrund rücken Large Language Models (LLMs) als aufstrebende Technologie im Bereich Künstliche Intelligenz (KI) 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 [12, 34].

Besonders interessant sind in diesem Kontext europäische Open-Source-Modelle wie die von Mistral [3]. 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 [41].

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. [27], 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. [41] 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 Europäischen Union (EU) - im Vergleich zu internationalen außerhalb der EU entwickelten Modellen schlagen und welche Trade-offs dabei entstehen [41]. 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 tex-

tuellen Beschreibungen und dem Kontext erschlossen werden [39]. 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 [27], 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) [14] 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) [15].

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) [14]. 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** [14].
- 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 [8]. 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 [29].

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 [29]. 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 [29]. 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 [29]. Sie können auch personenbezogene Daten enthalten.
- **Datenassoziationen** (Eingangs- und Ausgangsassoziationen) verbinden Aktivitäten mit Datenobjekten und Datenspeichern und zeigen so Ein- und Ausgaben explizit an [29]. 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 [29]. 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 [29]. Sie können auf eine Übermittlung personenbezogener Daten an Dritte hinweisen (z.B. Transfer von Kundendaten an einen



Abbildung 2.2: Beispiel einer Datenassoziation als Datenschutzsinal.

externen Dienstleister).

- **Textannotationen und Assoziationen** dienen dazu, zusätzliche Informationen zu Prozessmodellen hinzuzufügen, die nicht durch die standardmäßigen BPMN-Elemente abgedeckt sind [29]. 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) [29]. Diese Darstellung enthält alle relevanten Strukturinformationen (Elementtypen, Namen, Beziehungen, Zuordnungen, Positionen der Elemente) und wird von vielen Prozess-Engines und Modellierungswerkzeugen wie Camunda [18] und BPMN.io [17] 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 [29]. 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. [44] 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 [5, 26, 32].

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 [24]. Ein deutlicher Vorteil aktueller LLMs ist Zero-/Few-Shot Learning. Damit lassen sich Aufgaben alleine über Instruktionen und wenige Beispiele lösen, ohne dass erneutes Training benötigt wird [7, 24]. 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 [24].
2. API-gestützte Mechanismen wie Function Calling oder Structured-Output/JSON-Mode mit Schemaüberprüfung [4, 30, 33].
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 [40].

Typische Fehlerbilder bei der Nutzung von LLMs sind Halluzinationen (plausibel wirkende, aber fehlerhafte Aussagen) und Formatfehler (wie z.B. ungültiges JSON). In [21] 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 [20].

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 [31]. Es ist ein multimodales Modell mit starken Text-, Bild- und Audiofähigkeiten. Proprietäre 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 [14].

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 [3], Deepseek [2] und Qwen [36].

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

Dieses Kapitel präzisiert die Aufgabe der Arbeit, die Qualitätsziele der Klassifikation und steckt den fachlichen Geltungsbereich ab. Außerdem wird das Experimentdesign beschrieben, um die Forschungsfragen systematisch zu beantworten. Damit schafft es die Grundlage für die in Kapitel 4 beschriebene Klassifizierungspipeline sowie für die späteren Experimente und deren Auswertung.

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) [29] 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 [29].
- **Ausgabe** ist eine Menge von Aktivitäts-ids, die als *kritisch* klassifiziert wurden. Optional werden zusätzlich eine natürlichsprachige Begründung für einzelne Entscheidungen ausgegeben. Im Fall der Klassifizierungspipeline dieser Arbeit werden die Begründungen vom LLM generiert. Die Erklärungen dienen ausschließlich 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. Personenbezogene Daten sind, nach Art. 4 Abs. 1 DSGVO [14], alle

Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person beziehen. Gemäß Art. 4 Abs. 2 DSGVO [14] umfasst Verarbeitung jede mit personenbezogenen Daten vorgenommene Operation, wie z. B. Erheben, Speichern, Abrufen, Verwenden, Übermitteln und Löschen. Dies schließt auch die *Nutzung bereits vorhandener Daten* (z. B. Lesen/Abgleichen) ein.

Diese Aufgabenstellung reiht sich in Arbeiten zur Kennzeichnung kritischer/unkritischer Tätigkeiten in Prozessartefakten ein und bildet die Referenz für die Qualitätsziele im nächsten Abschnitt. [27]

3.2 Qualitätsziele

Die Aufgabe der datenschutzrechtlichen Klassifikation von Prozessen ist risikosensitiv. Übersehene kritische Aktivitäten, auch False Negatives (FN) genannt, bergen erhebliche Compliance-Risiken und können zu hohen Strafen nach der DSGVO führen. Beispielsweise erhielt Meta Platforms Ireland Limited (Meta IE) 2023 aufgrund von rechtswidriger Übermittlung von EU Nutzerdaten in die USA eine Geldbuße von 1,2 Milliarden Euro [1]. Auch Amazon wurde 2025 nach einem langjährigen Rechtsstreit wegen Datenschutzverstößen mit 746 Millionen Euro bestraft [13, 38]. Um derartige Strafen zu vermeiden, müssen kritische Aktivitäten zuverlässig identifiziert werden. Daher ist das **Hauptziel** der Klassifikation:

Maximaler Recall bei *minimalen FN* und zugleich *akzeptabler Precision*, damit der manuelle Prüfaufwand durch False Positives (FP) begrenzt bleibt.

Konfusionsmatrix und Metriken

Zur Bewertung des Hauptziels wird eine Konfusionsmatrix verwendet. Im vorliegenden binären Kontext entspricht die positive Klasse DSGVO-kritischen Aktivitäten. Die vier Felder der Konfusionsmatrix haben folgende Bedeutung [42]:

True Positives (TP) sind korrekt als kritisch erkannte Aktivitäten. Sie bilden den unmittelbaren *Nutzen* der Klassifikation.

FP sind fälschlich als kritisch markierte Aktivitäten. Sie erhöhen den manuellen Prüfaufwand, verursachen aber *keine* unmittelbaren Compliance-Risiken.

True Negatives (TN) sind korrekt als unkritisch eingestufte Aktivitäten und reduzieren den Gesamtaufwand.

FN sind übersehene kritische Aktivitäten. Sie sind besonders problematisch [27], da sie zu ausbleibender Risikobehandlung und potenziellen Bußgeldern führen können.

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

Diese Metriken sind in Information Retrieval und Maschinellem Lernen seit langem etabliert und bilden den De-facto-Standard zur Bewertung von Klassifikatoren [25, 42, 27]. Für das hier betrachtete binäre Problem gelten:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}, \quad \text{Precision} = \frac{TP}{TP + FP},$$
$$\text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Zielwerte

Vor dem Hintergrund der oben definierten Metriken und dem Hauptziel werden im Folgenden mithilfe von vergleichbarer Literatur realistische Zielkorridore abgeleitet.

Ähnliche Arbeiten, wie von Nake et al. [27], 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 [19] und F1-Scores von $\approx 0,68$ bis zu $\approx 0,79$ [41].

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.
- **FP je Prozess** sollen im Mittel $\leq 1,5$ betragen.

Nake et al. [27] 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.

Zusammenfassend fixieren die Zielwerte die angestrebte Performance. Im nächsten Abschnitt wird dargelegt, dass aufgrund der nicht-deterministischen Natur von LLMs die Ergebnisstabilität über wiederholte Läufe berücksichtigt werden muss.

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. [37] 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 vorgeschlagen, 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 Wiederholungen berichtet. Es 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 und 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 sinnvoller Kontext (z. B. Prozessname, Pool/Lane, Datenobjekte) berücksichtigt.
- Labels und Artefakte liegen in Deutsch und Englisch vor.
- Es handelt sich um ein *Screening*, nicht um eine Rechtsprüfung. Kritisch klassifizierte Aktivitäten sind anschließend juristisch zu prüfen.

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 [27].
- Optional generierte LLM-Begründungen sind als *Hilfetexte* zu verstehen, um die Entscheidung des LLMs 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.

3.4 Experimentdesign

Das gesamte Kapitel definierte die binäre Klassifikation von BPMN-Aktivitäten als kritisch/unkritisch mit Fokus auf maximalen Recall bei akzeptabler Precision und legte Qualitätsziele, Metriken, Geltungsbereich sowie Annahmen fest. Darauf aufbauend beschreibt dieser Abschnitt das Experimentdesign, mit dem LLMs fair und reproduzierbar verglichen werden, um die Forschungsfrage **FF1** sowie die Unterfragen **UF1–UF4** zu beantworten. Die konkrete Ausgestaltung und Durchführung der Experimente werden in Kapitel 8 *Versuchsaufbau* erläutert. Im Folgenden werden die wesentlichen Aspekte des Experimentdesigns beschrieben:

Ziel Ziel ist ein transparenter Vergleich mehrerer LLMs, die alle dieselbe Klassifizierungspipeline durchlaufen. Sie wird in Kapitel 4 daher so entworfen, dass sich das LLM austauschen lässt. Damit das Evaluationsframework, das in Kapitel 6 beschrieben wird, verschiedene Modelle unmittelbar gegeneinander ausführen kann, erfolgt der Wechsel der LLMs zur Laufzeit über übergebene Parameter, die das Modell definieren.

Vergleichsgegenstand Die Experimente werden über eine deklarative Konfiguration definiert, siehe Kapitel 6.3. Sie legt fest, welche Modelle, Datensätze und weitere Parameter zum Einsatz kommen. Je nach Auswahl werden mehrere Modelle und Modellvarianten parallel im Evaluationsframework ausgeführt, darunter Open-Source- und kommerzielle Modelle. Die deklarative Konfiguration sorgt für Portabilität und Wiederholbarkeit.

Datenbasis Als Datenbasis dienen die im Labeling-Tool erzeugten, gelabelten Testdatensätze, siehe Kapitel 5. Ein Testdatensatz enthält mehrere gelabelte Testfälle. Ein Testfall umfasst ein BPMN-Prozessmodell mit Labeln, die als DSGVO-kritischen Aktivitäten markieren. Die Auswahl der Datensätze für ein Experiment erfolgt in der Evaluierungskonfiguration, das Laden der Testfälle während der Laufzeit. Die Datensätze sollten idealerweise unterschiedliche Eigenschaften abdecken, damit die Forschungsfrage und die Unterfragen möglichst umfassend beantwortet werden.

Metriken und Erfolgskriterium Ausgewertet werden die in Abschnitt 3.2 beschriebenen Metriken: Accuracy, Precision, Recall und F1. Zusätzlich werden die Kennzahlen der Konfusionsmatrix betrachtet: TP, FP, TN, FN. Ein Testfall gilt

als *bestanden*, wenn die vom Modell als kritisch ausgegebenen Aktivitäten exakt den gelabelten kritischen Aktivitäten entsprechen. Technische Fehler werden separat ausgewiesen.

Ablauf eines Experiments

Ein Experiment verläuft in folgenden Schritten:

1. **Konfiguration laden.** Die Konfiguration mit Modellen, Datensätzen und optionalem seed wird geladen.
2. **Ausführung.** Für jedes Modell werden alle ausgewählten Testfälle durch die Klassifizierungspipeline verarbeitet. Pro Testfall werden TP, FP, FN, TN sowie der Status „bestanden“ oder „nicht bestanden“ berechnet.
3. **Stabilität.** Die Läufe erfolgen mit temperature gleich 0¹ und festem seed, sofern das jeweilige LLM dies unterstützt. Um die Nicht-Deterministik moderner LLMs abzubilden, werden die Experimente mehrfach mit unterschiedlichen Seeds wiederholt. Die Ergebnisse werden über die Läufe gemittelt.
4. **Bericht.** Aggregierte Kennzahlen pro Modell, wie Konfusionsmatrix, die genannten Metriken sowie die Bestehensraten werden ausgegeben. Metadaten wie verwendete Modelle, Datensätze und Seeds werden dokumentiert.

Dieses Kapitel definiert, *was* verglichen wird: Modelle, Datensätze und Metriken. Es beschreibt zudem, *wie* der Vergleich erfolgt. Kapitel 8 dokumentiert später die praktische Umsetzung mit konkreten Modellen, exakten Parameterwerten, Seeds sowie den vollständigen genutzten Konfigurationen. Im nächsten Kapitel folgt das Design und die Implementierung der Klassifizierungspipeline, die für den Vergleich der LLMs verwendet wird.

// TODO Kapitel Versuchsaufbau und Durchführung können durch die Existenz von diesem Kapitel wahrscheinlich auch gut zusammengefasst werden in einem Kapitel, so wie ich es auch in diesem Abschnitt geschrieben habe.

¹Die temperature steuert die Zufälligkeit der Textgenerierung bei LLMs. Niedrige Werte liefern stabilere Antworten, hohe Werte vielfältigere, jedoch weniger verlässliche [28].

4 Design und Implementierung der Klassifizierungspipeline

Dieses Kapitel beschreibt die Pipeline zur Klassifikation DSGVO-kritischer Aktivitäten in BPMN-Prozessen. Ausgehend von der in Kapitel 3.1 formulierten Aufgabenstellung wird der gesamte Weg von der Eingabe eines *BPMN-XML* über die Vorverarbeitung, das Prompt Engineering bis hin zur strukturierten, schema-konformen Ausgabe aufgezeigt. Außerdem wird ein HTTP-basiertes API-Design vorgestellt, das die Einbindung in weitere Werkzeuge und das Evaluationsframework ermöglicht. Der Prozessfluss der Klassifizierungspipeline ist in Abbildung 4.1 dargestellt und wird in diesem Kapitel im Detail erläutert.

Die Klassifizierungspipeline soll eine binäre Entscheidung auf Ebene einzelner BPMN-Aktivitäten treffen: Für jede Aktivität eines Eingabemodells wird bestimmt, ob sie *kritisch* im Sinne der DSGVO ist. Die Pipeline ist so konzipiert, dass sie mit Modellen aus gängigen Modellierungswerkzeugen kompatibel ist.

4.1 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 [9, 10]. Auf dieser Basis werden die relevanten Informationen extrahiert und in der Datenklasse `BpmnElement` strukturiert abgelegt. Die Datenklasse ist in Listing 4.1 zu sehen.

4 Design und Implementierung der Klassifizierungspipeline

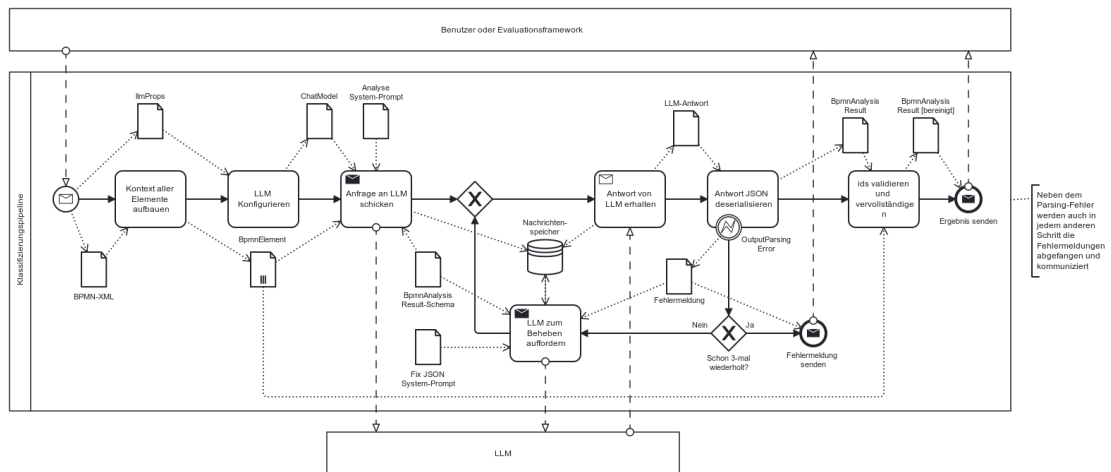


Abbildung 4.1: BPMN-Diagramm der Klassifizierungspipeline.

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. In Abbildung 4.1 ist dieser Schritt über die Aktivität „Kontext aller Elemente aufbauen“ dargestellt.

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

```

1 data class BpmnElement(
2     val type: String,
3     val id: String,
4     val name: String? = null,
5     val documentation: String? = null,
6     val poolName: String? = null,
7     val laneName: String? = null,
8     val outgoingFlowElementIds: List<String> = emptyList(),
9     val incomingFlowElementIds: List<String> = emptyList(),
10    val outgoingMessageFlowsToElementIds: List<String> = emptyList(),
11    val incomingMessageFlowsFromElementIds: List<String> = emptyList()
12    ↪ ,
13    val incomingDataFromElementIds: List<String> = emptyList(),
14    val outgoingDataToElementIds: List<String> = emptyList(),
15    val associatedElementIds: List<String> = emptyList()

```

4.2 Prompt Engineering

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* [23] benutzt. 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 Rückgabetypp deserialisiert [35]. Beim erstellen des AI Service werden ein `ChatModel`, die Systemnachricht und die Interface-Methoden konfiguriert. Ein `ChatModel` ist die spezifische Implementierung der Chat-Completion-Schnittstelle eines LLM von Langchain4j [22]. Die Methodenparameter der Interface-Methoden repräsentieren die Nutzereingabe. Der Rückgabetypp 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 [35].

Die Kommunikation mit dem LLM erfolgt damit über einfache Funktionsaufrufe, während LangChain4j Prompt-Erzeugung, Parameterbindung sowie die Deserialisierung der Antwort übernimmt [35]. 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 Ein- und Ausgabe-paaren pro Prozess enthalten. Zero-Shot reduziert den Pflegeaufwand und nutzt die In-Context-Fähigkeiten moderner Modelle, nur über Instruktionen zu generalisieren [7, 24]. 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.

Die Kombination dieser Elemente im System-Prompt stellt sicher, dass das LLM die Aufgabe versteht, die relevanten Kriterien kennt und die Ausgabe in einem maschinenlesbaren Format liefert. So entsteht die Basis für eine zuverlässige Klassifikation. Zu einer Anfrage an ein LLM gehört außerdem stets ein User-Prompt, der die eigentliche Nutzereingabe enthält. Dessen Aufbau wird im nächsten Abschnitt beschrieben.

User-Prompt

Der User-Prompt übergibt dem LLM die konkreten Eingabedaten einer Anfrage. Während der System-Prompt Regeln, Ziele und Ausgabevorgaben festlegt, liefert der User-Prompt die Fall- bzw. Kontextinformationen, auf die diese Regeln angewendet werden.

Der User-Prompt wird mithilfe der Daten aus der Vorverarbeitung aus Abschnitt 4.1 erzeugt und enthält eine Liste von `BpmnElement`-Objekten, siehe Listing 4.1. Die Interaktion mit dem LLM erfolgt deklarativ über *LangChain4j*. Dafür wird die Liste der `BpmnElement`-Objekte als Methodenparameter mit der Annotation `@UserMessage` an die Interface-Methode übergeben (Listing 4.2) und dort automatisch in den User-Prompt eingebettet.

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

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

Zur Laufzeit serialisiert *LangChain4j* die *BpmnElement*-Liste zu einem JSON-Array und stellt sie als User-Prompt bereit. Der zuvor konfigurierte System-Prompt wird bei einer Anfrage an das LLM automatisch vor dem User-Prompt vorangestellt. Auf diese Weise wendet das LLM die im System-Prompt definierten Kriterien auf die im User-Prompt gelieferten Informationen zum BPMN-Prozessmodell an. Dadurch wird jede Aktivität des Prozesses genau so wie im System-Prompt beschrieben klassifiziert. In Abbildung 4.1 findet dieser Schritt in der Aktivität „Anfrage an das LLM schicken“ statt.

Zusammenfassend setzt der System-Prompt typischerweise das Regelwerk und der User-Prompt liefert die konkreten Eingabedaten. Besonders in mehrstufigen Dialogen mit dem LLM spielt dieses Muster eine größere Rolle, da der System-Prompt konstant bleibt, während der User-Prompt je nach Anfrage variiert. Im vorliegenden Szenario, wo immer nur genau eine Anfrage pro Prozessmodell gestellt wird fällt der Unterschied weniger ins Gewicht, als würden sämtliche Vorgaben direkt im User-Prompt stehen. Die Trennung erhöht dennoch die Nachvollziehbarkeit, sorgt für klare Rollen und erleichtert die Wiederverwendung.

Im folgenden Abschnitt wird beschrieben, wie auf dieser Basis strukturierte Ausgaben erzeugt werden, damit im Code direkt mit typisierten Objekten weitergearbeitet werden kann.

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 Paaren aus *id*, *reason* und *isRelevant*. Siehe A.2 für die vollständige Definition der Datenklasse. *Langchain4j* inferiert auf Basis des Rückgabetyps der Interface-Methode ein JSON-Schema und fügt dieses automatisch dem User-Prompt zusammen mit der Aufforderung hinzu, die Antwort in diesem JSON-Format zu liefern [35]. Durch die explizite Angabe des gewünschten JSON-Formats im Prompt wird die Format-Treue der Antwort erhöht

[24], also die Wahrscheinlichkeit, dass die Antwort tatsächlich dem gewünschten Schema entspricht.

Einige LLMs unterstützen darüber hinaus die Möglichkeit, das Antwortformat API-seitig zu erzwingen. Das ist beispielsweise bei Mistral und OpenAI der Fall [4, 33]. Falls das LLM die `response_format` Funktionalität unterstützt setzt LangChain4j dies zusätzlich auf das gewünschte Schema und erzwingt so das Ziel-JSON API-seitig [35]. Fehlt diese Fähigkeit, greift ausschließlich die Prompt-basierte Schemaanweisung.

Das vom LLM gelieferte JSON deserialisiert *LangChain4j* anschließend automatisch zu einem `BpmnAnalysisResult`. So kann im Code direkt mit einem typsicheren Objekt weitergearbeitet werden. In Abbildung 4.1 ist dieser Prozess über die Aktivitäten „Antwort von LLM erhalten“ und „Antwort JSON deserialisieren“ dargestellt.

4.3 Validierung der Ausgabe

Zusätzlich zu den in Kapitel 4.2 beschriebenen Maßnahmen stellt die Pipeline mehrere Validierungs- und Korrekturschritte bereit, die in Abbildung 4.1 direkt auf „Antwort JSON deserialisieren“ folgen. Diese Schritte dienen dazu, die Qualität und Korrektheit der Ausgabe des LLM zu gewährleisten. Im Folgenden werden die einzelnen Validierungsmechanismen erläutert.

Schema-Parsing und Retry-Mechanismus

Die vom LLM zurückgelieferte Antwort wird zunächst von Langchain4j zu einem `BpmnAnalysisResult` deserialisiert. Entspricht die Struktur dabei nicht dem erwarteten JSON-Schema, löst Langchain4j eine `OutputParsingException` aus. In diesem Fall greift der in Abbildung 4.1 ab dem Boundary-Error-Event „Output-ParsingError“ dargestellte Retry-Mechanismus. Dabei wird bis zu dreimal die ursprüngliche Anfrage erneut gesendet, ergänzt um die Parser-Fehlermeldung sowie eine explizite Anweisung, die Ausgabe exakt gemäß Schema zu formatieren. So bleiben sowohl der Kontext der ursprünglichen Anfrage als auch die Information

über den aufgetretenen Fehler erhalten, damit das LLM die Ausgabe entsprechend anpassen kann. Schlagen alle drei Versuche fehl, wird der Fehler an die aufrufende Schnittstelle zurückgegeben und die Klassifizierung gilt als fehlgeschlagen.

Ein zusammenfassender Log-Auszug des Retry-Mechanismus findet sich in Listing A.5. Er zeigt exemplarisch, dass zunächst der boolesche Wert `isRelevant` fehlt und im zweiten Versuch korrekt ergänzt wird.

Relevanz-Filterung

Nach erfolgreichem Parsing werden alle Elemente mit `isRelevant = false` entfernt. Dieser Schritt geschieht bereits im Konstruktor der Datenklasse `BpmnAnalysisResult` automatisch. Dieser Mechanismus adressiert modellseitige *Überklassifizierungen*, bei denen das LLM fälschlicherweise `ids` von Aktivitäten ausgibt, obwohl sie nicht DSGVO-kritisch sind. Es wird sichergestellt, dass nur Aktivitäten, die als kritisch klassifiziert wurden, in der finalen Ausgabe verbleiben.

Ohne das `isRelevant`-Flag hat das LLM in der Praxis des Öfteren Aktivitäten als kritisch ausgegeben, deren Begründung jedoch ausdrücklich darlegte, *warum* sie *nicht* kritisch seien. Das Modell erkannte die Unkritikalität also korrekt, hielt sich aber nicht strikt an die Vorgabe, ausschließlich `ids` kritischer Aktivitäten in die Antwort aufzunehmen. Als pragmatische Absicherung wurde daher das boolesche `isRelevant`-Flag eingeführt. Das LLM muss zusätzlich neben der Ausgabe der `ids` auch explizit angeben, ob die jeweilige Aktivität kritisch ist oder nicht. In der Summe reduziert diese Filterung die Anzahl widersprüchlicher Ausgaben.

`isRelevant` dient ausschließlich einer internen Validierung und wird in der finalen Ausgabe der Klassifizierungs-Pipeline nicht berücksichtigt.

id-Validierung und -Vervollständigung

In der Praxis liefert das LLM mitunter unvollständige oder fehlerhafte `id`-Werte, die im Prozess nicht existieren. Zur Erhöhung der Robustheit werden die vom LLM ausgegebenen `ids` daher gegen die tatsächlich im Prozess vorhandenen Aktivitäts-`ids` geprüft und – wenn möglich – automatisch vervollständigt. Der Ablauf ist:

1. Ermittlung der Grundmenge aller gültigen Aktivitäts-ids aus der `BpmnElement`-Liste, die beim Preprocessing erstellt wurde.
2. Für jede vom LLM gelieferte `id` wird ein Präfix-Match gegen die gültigen `ids` durchgeführt. Ist die ausgegebene `id` Präfix *genau einer* gültigen `id`, wird sie durch diese vollständige `id` ersetzt.
3. Bleibt das Präfix-Match ohne eindeutiges Ergebnis, folgt ein Substring-Match: Ist die ausgegebene `id` Teilstring *genau einer* gültigen `id`, wird entsprechend vervollständigt.
4. Liefert weder Präfix- noch Substring-Match eine eindeutige Übereinstimmung, gilt die ausgegebene `id` als ungültig und wird aus der finalen Ausgabe entfernt.
5. Abschließend werden Duplikate entfernt, sodass jede kritische Aktivität höchstens einmal in der Ausgabe erscheint.

Gibt das LLM beispielsweise die `id` `Activity_1` aus, existiert im Prozess jedoch nur `Activity_12345`, wird die Ausgabe automatisch auf die korrekte `id` vervollständigt. Existieren hingegen sowohl `Activity_123` als auch `Activity_124` im Prozess, bleibt die Ausgabe unvollständig und wird entfernt, da keine eindeutige Zuordnung möglich ist.

Dieser Schritt fängt typische LLM-Ausgabefehler ab – etwa Halluzinationen oder abgeschnittene Bezeichner – und stellt die Konsistenz mit dem Eingabemodell sicher. Im Diagramm 4.1 ist er als „ids validieren und vervollständigen“ markiert. Ein fokussierter Code-Auszug findet sich in Listing A.3.

Zusammenfassung

Nach der Validierung besteht `BpmnAnalysisResult` nur noch aus Aktivitäten, die vom LLM als kritisch eingestuft wurden (`isRelevant = true`) und deren `ids` im Prozess existieren.

Da es nun möglich ist, BPMN-Prozesse vorzuverarbeiten, zu klassifizieren und die Ausgabe zu validieren und zu beheben, folgt als nächster Schritt die Definition einer Schnittstelle zum Aufruf der Pipeline. Das nächste Kapitel beschreibt dafür das API-Design.

4.4 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 6) 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 im Folgenden definiert wird. 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 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",
4  "type": "object",
5  "properties": {
6    "baseUrl": { "type": "string" },
7    "modelName": { "type": "string" },
8    "apiKey": { "type": "string" },
9    "timeoutSeconds": { "type": "number" },
10   "seed": { "type": "number" }
11 },
12 "required": []
13 }
```

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.

```
1  {
2    "$schema": "https://json-schema.org/draft/2020-12/schema",
3    "title": "BpmnAnalysisResult",
4    "type": "object",
5    "properties": {
6      "criticalElements": {
7        "type": "array",
8        "items": {
9          "type": "object",
10         "properties": {
11           "id": { "type": "string" },
12           "name": { "type": "string" },
13           "reason": { "type": "string" }
14         },
15         "required": ["id"]
16       }
17     }
18   },
```

```
19   "required": ["criticalElements"]
20 }
```

Ein Beispielaufwurf 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: Beispielaufwurf des Klassifizierungsendpunkts mit `curl`.

```
1 curl -X POST http://localhost:8080/gdpr/analysis/prompt-engineering \
2   -F 'bpmnFile=@/path/to/process.bpmn' \
3   -F 'llmProps={
4       "baseUrl": "https://api.mistral.ai/v1",
5       "modelName": "mistral-small-latest",
6       "apiKey": "*****"
7   }'
```

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 6). 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.

4.5 Nutzung über Webapp

Zur interaktiven Nutzung der Klassifizierung wurde eine *Sandbox* in Form einer Webapp entwickelt. Sie verbindet einen vollwertigen BPMN-Editor auf Basis von

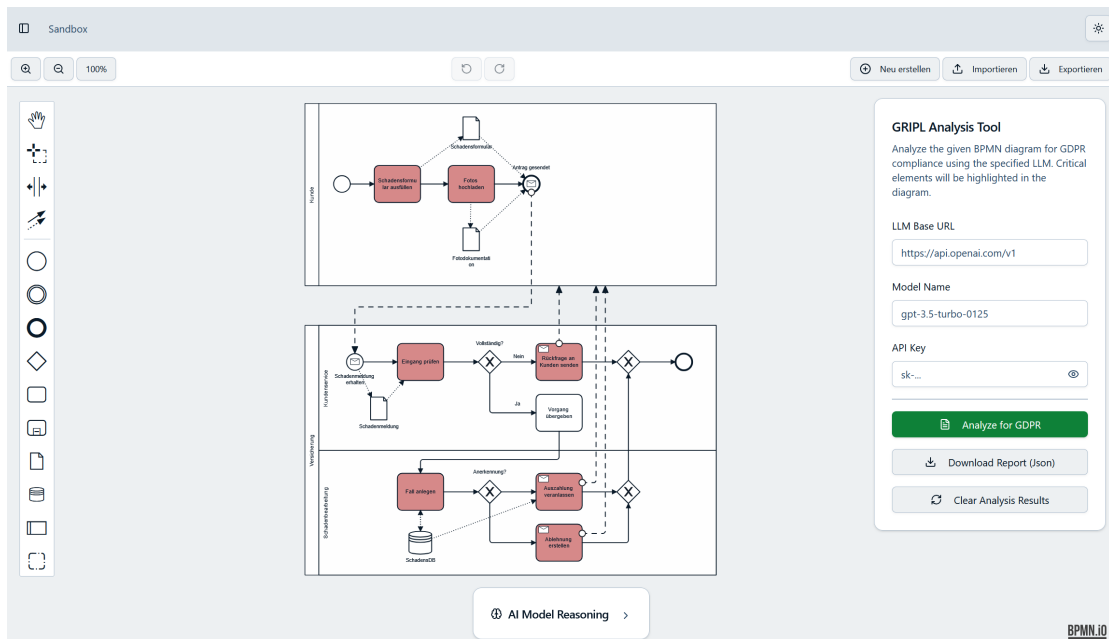
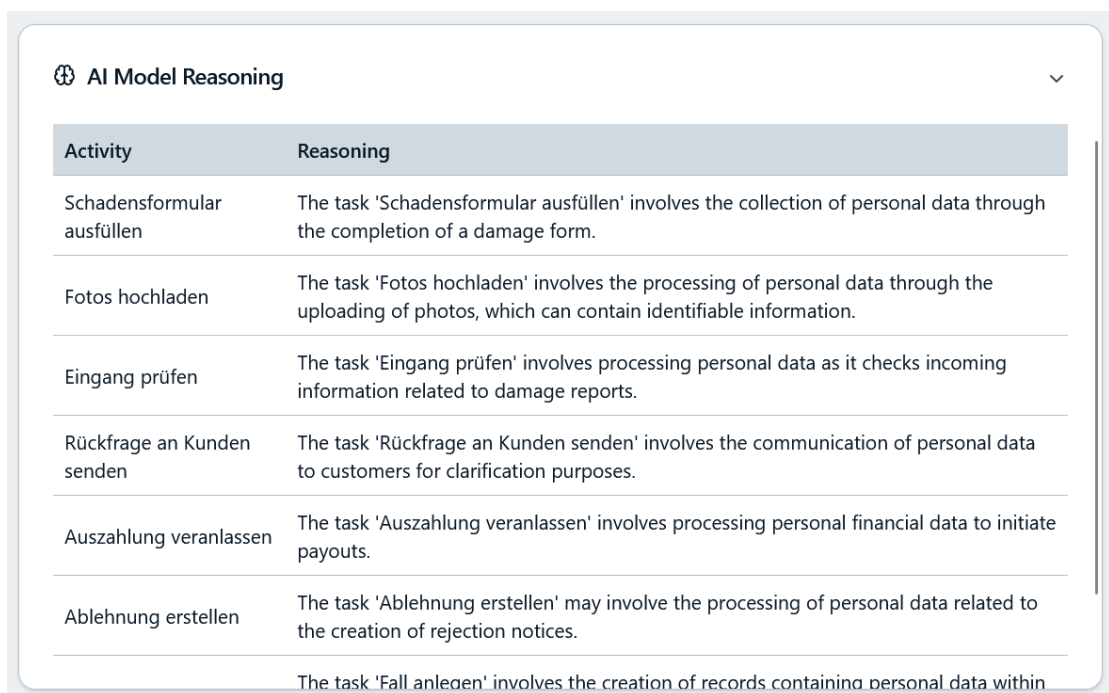


Abbildung 4.2: Sandbox im Frontend mit hervorgehobenen kritischen Aktivitäten nach Analyse.

BPMN.js [16] mit der in Kapitel 4.4 beschriebenen HTTP-Schnittstelle und macht die Analyse damit intuitiv bedienbar. In der Sandbox können BPMN-Modelle erstellt, verändert, exportiert und importiert sowie auf Datenschutzrelevanz analysiert werden. Als kritisch klassifizierte Aktivitäten werden nach der Analyse direkt im Editor farblich hervorgehoben, wie in Abbildung 4.2 zu sehen ist.

Außerdem können die vom LLM generierten Begründungen zu jeder als kritisch erkannten Aktivität im Editor eingesehen werden. Diese Erläuterungen werden gesammelt in einer aufklappbaren Karte im unteren Bereich des Editors angezeigt, siehe Abbildung 4.3.

Um verschiedene LLMs vergleichen zu können, verfügt die Sandbox auf der rechten Seite über ein Einstellungsmenü mit konfigurierbaren LLM-Parametern (siehe Abbildung 4.2). Diese Parameter sind identisch zu den in Kapitel 4.4 beschriebenen `LLMProps` und werden beim Starten der Analyse in die API-Anfrage überführt.



The screenshot shows a user interface titled "AI Model Reasoning" with a dropdown arrow. Below the title is a table with two columns: "Activity" and "Reasoning". The table lists six activities with their corresponding reasoning explanations. At the bottom of the table, there is a partial entry for "Fall anlegen".

Activity	Reasoning
Schadensformular ausfüllen	The task 'Schadensformular ausfüllen' involves the collection of personal data through the completion of a damage form.
Fotos hochladen	The task 'Fotos hochladen' involves the processing of personal data through the uploading of photos, which can contain identifiable information.
Eingang prüfen	The task 'Eingang prüfen' involves processing personal data as it checks incoming information related to damage reports.
Rückfrage an Kunden senden	The task 'Rückfrage an Kunden senden' involves the communication of personal data to customers for clarification purposes.
Auszahlung veranlassen	The task 'Auszahlung veranlassen' involves processing personal financial data to initiate payouts.
Ablehnung erstellen	The task 'Ablehnung erstellen' may involve the processing of personal data related to the creation of rejection notices.
Fall anlegen	The task 'Fall anlegen' involves the creation of records containing personal data within

Abbildung 4.3: Begründung der Klassifikation durch das LLM in der Sandbox.

5 Labeling und Datensätze

Für die Evaluation der Klassifikation ist es nötig, zuvor entsprechende Testdatensätze mit Annotationen bereitzustellen. Ein solcher Datensatz besteht dabei aus mehreren Testfällen, wobei jeder Testfall ein BPMN-Prozessmodell darstellt. Standardisierte Datensätze gewährleisten einheitliche Prüfbedingungen und ermöglichen so objektive Leistungsvergleiche.

5.1 Labeling Tool

Um die Erstellung und Verwaltung von gelabelten BPMN-Prozessmodellen zu erleichtern, wurde eine Webapplikation entwickelt. Mit dieser können BPMN-Testfälle erstellt, bearbeitet und Aktivitäten mit Labels versehen werden. Wichtige Funktionen des Labeling-Tools umfassen:

- Anlegen und Verwalten von Datensätzen.
- Erstellung beliebig vieler Testfälle pro Datensatz.
- Direkte Bearbeitung von BPMN-Modellen im Browser mittels BPMN.io [17].
- Labeling-Modus, in dem Aktivitäten als DSGVO-kritisch markiert werden können. Optional kann eine Begründung für die Markierung angegeben werden.
- Persistente Speicherung der annotierten Testfälle in einer Datenbank für die spätere Nutzung im Evaluationsframework (siehe Kapitel 6).

Abbildung 5.1 zeigt den Labeling-Editor: Hier können Anwender Prozessmodelle erstellen und Aktivitäten im Labeling-Modus direkt als kritisch markieren. Optional kann für jede markierte Aktivität eine Begründung eingegeben werden. Diese

5 Labeling und Datensätze

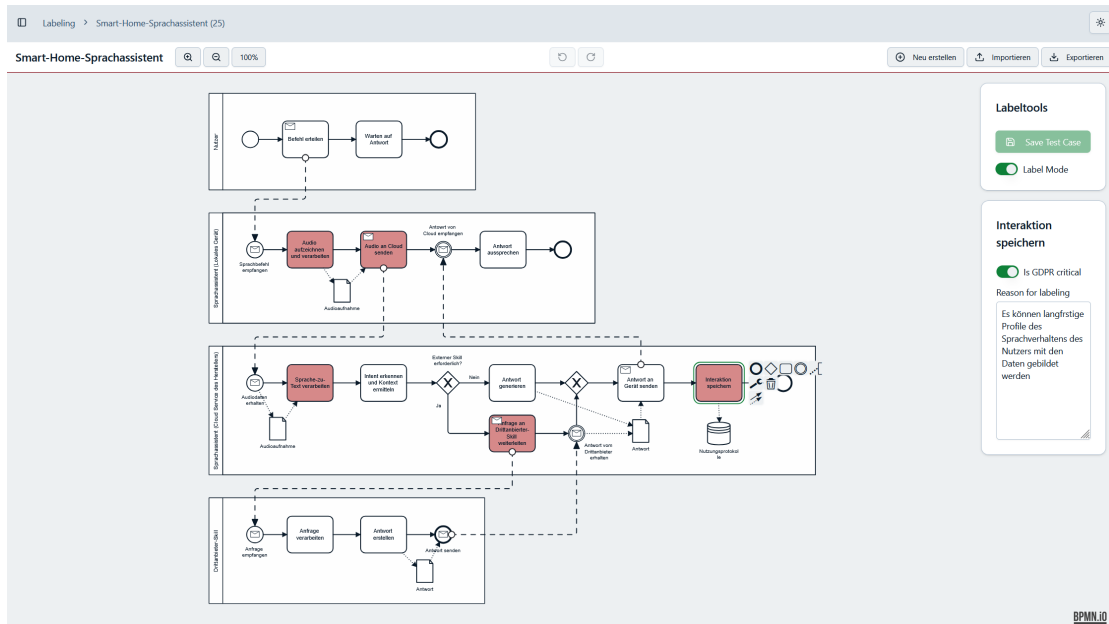


Abbildung 5.1: Labeling-Editor im Labeling-Modus mit exemplarischem Modell.

Begründung ist ausschließlich als Hilfe für den Anwender gedacht, um die eigene Entscheidung zu dokumentieren. Die Begründung wird nicht in der Evaluierung berücksichtigt.

In der Übersicht der Datensätze aus Abbildung 5.2 sind alle angelegten Datensätze und zugehörigen Testfälle aufgelistet. Von hier aus können neue Datensätze und Testfälle erstellt sowie bestehende bearbeitet werden.

5.2 Quellen und Eigenschaften der Datensätze

Für die Evaluation wurden drei Gruppen von BPMN-Datensätzen eingesetzt:

1. Prozesse, die von der Universität Ulm bereitgestellt wurden (z.B. Lehrbeispiele aus Übungsaufgaben).
2. Realistische, mittelgroße Szenarien aus verschiedenen Domänen. Diese Prozesse beinhalten Elemente wie Pools, Lanes, Datenobjekte und Gateways.
3. Kleine, reduzierte Testfälle mit maximal fünf Aktivitäten und wenigen weiteren Elementen (z.B. einfacher Sequenzfluss ohne Pools).

5 Labeling und Datensätze

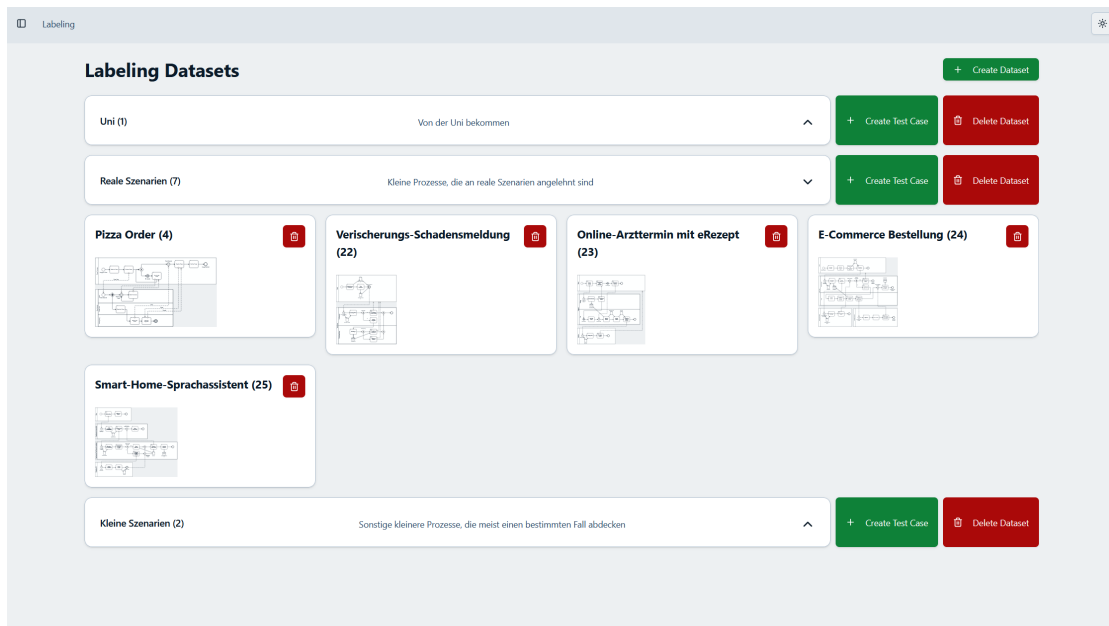


Abbildung 5.2: Übersicht der Datensätze im Labeling-Tool.

Diese heterogene Auswahl ist bewusst getroffen worden, da die Mischung aus verschiedenen Domänen und Modellkomplexitäten eine aussagekräftige Evaluation ermöglicht. In der Literatur wird betont, dass eine erhöhte Datensatzvielfalt die Robustheit der Bewertung steigert und einseitige Ergebnisse vermeidet [6]. Tabelle 5.1 zeigt die Eckdaten der Datensätze.

// TODO Noch mehr Testfälle hinzufügen - insbesondere, um auf die Datenassoziationsthematik aus dem Kapitel 2.2 einzugehen. Ich sollte also mindestens ein kleines Beispiel jeweils mit und ohne Datenassoziation haben, um den Unterschied zu verdeutlichen. Das bedeutet aber auch, dass ich die Tabelle 5.1 neu berechnen muss.

// TODO Hier noch einen Abschnitt wo ich ein paar besondere Testfälle hervorhebe, z.B. mit speziellen Datenassoziationen, oder reicht der Überblick mit den Eckdaten? Spätestens in den Fallstudien der Ergebnisse werde ich ja nochmal auf einzelne Testfälle eingehen.

Tabelle 5.1: Eckdaten der verwendeten Datensätze

	Uni-Prozesse	Reale Szenarien	Kleine Testfälle
Testfälle Gesamt	5	5	10
Testfälle (DE)	0	4	10
Testfälle (EN)	5	1	0
Ø Aktivitäten \pm SD ¹	13,4 \pm 2,6	11,6 \pm 4,2	5 \pm 0
Ø Aktivitäten (kritisch) \pm SD	8,6 \pm 3,6	6,6 \pm 1,9	2,6 \pm 1,5
Ø Datenobjekte \pm SD	1,4 \pm 1,9	3,6 \pm 2,1	0 \pm 0
Ø Datenassoziationen \pm SD	2,4 \pm 3,3	7 \pm 4	0 \pm 0
Ø Ereignisse \pm SD	21 \pm 13,8	8,2 \pm 2,8	2 \pm 0
Ø Gateways \pm SD	13 \pm 7,6	1,8 \pm 1,5	0 \pm 0
Ø Pools \pm SD	3,4 \pm 1,1	3 \pm 1	0 \pm 0
Ø Lanes \pm SD ²	3 \pm 1	4 \pm 0,7	0 \pm 0
Ø Nachrichtenflüsse \pm SD	9,4 \pm 5,3	5,2 \pm 0,8	0 \pm 0
Ø Annotationen \pm SD	1 \pm 1,7	0 \pm 0	0 \pm 0

¹ SD = Standardabweichung s der jeweiligen Anzahl pro Testfall.

² Blackbox-Pools ohne Lanes wurden nicht mitgezählt, daher kann der Durchschnittswert der Lanes geringer ausfallen als der, der Pools.

5.3 Labeling-Guide

Die Aktivitäten in den Testfällen sollen mit dem Label „kritisch“ versehen werden, wenn sie potenziell personenbezogene Daten verarbeiten und somit im Sinne der DSGVO relevant sein könnten. Die wichtigsten Begriffe der DSGVO wurden bereits in Abschnitt 2.1 definiert.

Beim Labeln einer Aktivität können Grenzfälle auftreten – etwa wenn kein Datenobjekt vorhanden ist, der Name aber auf Datenverarbeitung hindeutet (z. B. „Verträge archivieren“). Solche Verträge können personenbezogen sein (z. B. Arbeitsverträge) oder rein geschäftlich zwischen Unternehmen. In diesen Fällen wird zunächst der Kontext geprüft: Gibt es Hinweise auf personenbezogene Daten (z.,B. Pool/Lane oder angrenzende Aktivitäten im Prozess)? Fehlen eindeutige Hinweise, wird die Aktivität als unkritisch gelabelt. Deutet der Kontext hingegen auf die Verarbeitung personenbezogener Daten hin (z.,B. ein Prozessname wie „Mitarbeiterverwaltung“ oder vorangehende Aktivitäten wie „Mitarbeiterdaten erfassen“), erhält die Aktivität das Label kritisch. Im Zweifel wird kritisch gelabelt, um eine höhere Sensitivität zu gewährleisten.

Tabelle 5.2 listet beispielhaft einige Aktivitäten mit ihrer Klassifikation und einer Begründung auf.

Tabelle 5.2: Beispielhafte Aktivitäten und Label

Aktivität	Kritisch?	Kommentar
Lieferadresse eingeben	Ja	Name, Anschrift des Kunden werden aufgenommen.
Rückfrage an Kunden senden	Ja	Kontaktinformationen werden verwendet.
Fall anlegen	Ja	Aktivität befindet sich im Kundenservice-Kontext, personenbezogene Daten wahrscheinlich.
Sprache zu Text verarbeiten	Ja	Im Kontext eines Sprachassistenten werden biometrische Daten des Nutzers verarbeitet.
Produkt versenden	Nein*	Logistik und Sachvorgänge sind nicht per se Datenschutzkritisch, solange keine neue Datenverarbeitung, wie ein Labeldruck stattfindet.
Systemprotokoll auslesen	Ja	Im Kontext einer technischen Wartung können personenbezogene Daten (z.B. Nutzer-ids) enthalten sein.
Logdaten archivieren (anonym)	Nein	Keine personenbezogenen Daten enthalten.
Gerät kalibrieren	Nein	Im Kontext einer technischen Wartung werden keine personenbezogenen Daten verarbeitet.

6 Evaluationsframework

Nachdem nun Daten gelabelt werden können und der Testdatensatz für diese Arbeit erstellt wurde, wird nun in diesem Kapitel das Evaluationsframework vorgestellt. Das Framework nutzt die in Kapitel 4 entwickelte Klassifizierungspipeline, um verschiedene LLMs anhand gelabelter Testdaten systematisch, reproduzierbar und transparent zu vergleichen. Leitendes Gestaltungsprinzip ist die Entkopplung: Modelle und Klassifizierungsalgorithmen werden zur Laufzeit konfiguriert und sind dadurch austauschbar. So ermöglicht das Framework einen fairen Vergleich unterschiedlicher Modelle und Verfahren.

6.1 Use-Cases und Anforderungen

Das Evaluationsframework richtet sich an Forschende und Entwickler, die LLMs und Klassifizierungsalgorithmen für die Identifikation DSGVO-kritischer BPMN-Aktivitäten auswerten und miteinander vergleichen möchten. Es bietet eine einheitliche Ausführungs- und Auswertungsumgebung mit klar definierten Schnittstellen und standardisierten Berichten. In diesem Kapitel werden die Use-Cases und Anforderungen des Evaluationsframeworks beschrieben.

Use-Cases

Die wichtigsten Anwendungsfälle des Evaluationsframeworks sind:

- **Benchmarking von LLMs.** Systematischer Vergleich mehrerer LLMs auf denselben Datensätzen, mit identischem Algorithmus und identischen Parametern.

- **A/B-Vergleich von Algorithmen.** Gegenüberstellung verschiedener Klassifizierungspipelines, mit z.B. alternativen Prompts oder anderem Preprocessing, über eine standardisierte HTTP-Schnittstelle, die in Kapitel 4.4 definiert ist.
- **Explorative Analyse.** Detaillierte Einsicht pro Modell und Testfall (inklusive Begründungen und Visualisierungen), um Fehlklassifikationen gezielt zu untersuchen.
- **Berichterstellung.** Die Ergebnisse lassen sich als JSON oder Markdown exportieren und später wieder importieren, um sie erneut untersuchen zu können. Sie eignen sich zudem für die Publikation. Die Diagramme werden automatisch erzeugt und stehen ebenfalls zum Download bereit.

In dieser Arbeit werden keine A/B-Vergleiche unterschiedlicher Klassifizierungsalgorithmen durchgeführt, sondern lediglich verschiedene LLMs mit demselben Algorithmus verglichen. Das Framework ist jedoch so konzipiert, dass dies in zukünftigen Arbeiten möglich ist.

Funktionale Anforderungen

In der folgenden Tabelle sind die funktionalen Anforderungen an das Evaluationsframework aufgelistet, die notwendig sind um Use-Cases zu erfüllen:

ID:	FA01
Titel:	Nutzen gelabelter Testdatensätze
Beschreibung:	Das Framework kann die gelabelten Testdatensätze benutzen, die mit dem Labeling-Tool aus 5.1 erstellt worden sind.
Abhängigkeit:	

ID:	FA02
Titel:	Vergleichbarkeit von Modellen und Algorithmen
Beschreibung:	Das Framework erlaubt den direkten Vergleich verschiedener LLMs sowie unterschiedlicher Klassifizierungsalgorithmen anhand gelabelter Testdaten. Die Anbindung an Klassifizierungsalgorithmen erfolgt über die in Kapitel 4.4 definierte, standardisierte HTTP-Schnittstelle.
Abhängigkeit:	FA01

ID:	FA03
Titel:	Deklarative Konfiguration
Beschreibung:	Ein Evaluationslauf ist vollständig über eine YAML-Datei konfigurierbar. Dazu zählen Modelle, Klassifizierungsendpunkte, Testdatensätze, Seed. Experimente werden dadurch portabel und wiederholbar.
Abhängigkeit:	FA02

ID:	FA04
Titel:	Detaillierte Ergebnisaufbereitung
Beschreibung:	<p>Das Framework gibt Ergebnisse auf zwei Ebenen aus.</p> <ol style="list-style-type: none"> 1. Pro Testfall und pro Modell: Status („bestanden“/„nicht bestanden“), klassifizierte Elemente mit Begründungen, TP/FP/FN/TN und eine Visualisierung der Klassifikation im BPMN-Prozess. 2. Pro Modell als Summe über alle Testfälle: Accuracy, Precision, Recall, F1-Score und die Konfusionsmatrix. <p>Zusätzlich protokolliert das Framework Metadaten der Evaluation, z. B. Endpunkt, verwendete Modelle und den Seed.</p>
Abhängigkeit:	FA02

ID:	FA05
Titel:	Frontend
Beschreibung:	Für eine einfache Bedienung und Ansicht der Ergebnisse bietet das Evaluationsframework ein Frontend an.
Abhängigkeit:	FA02, FA03, FA04

ID:	FA06
Titel:	Visualisierung und Berichte der Gesamtergebnisse
Beschreibung:	Kennzahlen werden als Side-by-Side-Diagramme und tabellarisch dargestellt. Zusätzlich stehen Export/Import der Ergebnisse als JSON sowie ein Markdown-Report zur Verfügung.
Abhängigkeit:	FA05

6.2 Testdaten

Wie in FA01 beschrieben, kann das Evaluationsframework die mit dem Labeling-Tool erzeugten Testdatensätze unmittelbar verwenden. Da das Tool die Testdaten in einer Datenbank ablegt, lassen sie sich unkompliziert auslesen und für die Evaluierung heranziehen. In der Konfiguration des Frameworks wird festgelegt, welche Datensätze genutzt werden, wodurch sich die Auswertung gezielt auf einen bestimmten Anwendungsfall zuschneiden lässt. Wie die Konfiguration einer Evaluierung funktioniert, wird im nächsten Kapitel erläutert.

6.3 Konfiguration einer Evaluierung

Die funktionale Anforderung FA03 fordert, dass Evaluationsläufe deklarativ konfiguriert werden können. Das Framework unterstützt dies auf zwei Wegen: Erstens bietet die Weboberfläche, die in 6.6 gezeigt wird, die Möglichkeit, Evaluationsläufe interaktiv zu konfigurieren und zu starten. Zweitens lässt sich eine Evaluierung über eine YAML-Datei beschreiben, die entweder in der Weboberfläche hochgeladen oder per CLI an das Evaluationsframework übergeben wird. Auf diese Weise werden Reproduzierbarkeit und Versionierung der Evaluationsläufe sichergestellt.

Listing 6.1 zeigt ein Beispiel für eine solche YAML-Konfiguration. Ein ausführliches JSON-Schema ist im Anhang (Listing A.4) zu finden.

Listing 6.1: Beispiel einer Evaluierungskonfiguration in YAML.

```
1 defaultEvaluationEndpoint: /gdpr/analysis/prompt-engineering
2 maxConcurrent: 10
3 seed: 42
4 models:
5   - label: Mistral Medium 3.1
6     llmProps:
7       baseUrl: https://openrouter.ai/api/v1
8       modelName: mistralai/mistral-medium-3.1
9       apiKey: ${OPEN_ROUTER_API_KEY}
10  - label: Deepseek Chat v3.1
11    llmProps:
12      baseUrl: https://openrouter.ai/api/v1
13      modelName: deepseek/deepseek-chat-v3.1
14      apiKey: ${OPEN_ROUTER_API_KEY}
15  - label: GPT oss 120b
16    llmProps:
17      baseUrl: https://openrouter.ai/api/v1
18      modelName: openai/gpt-oss-120b
19      apiKey: ${OPEN_ROUTER_API_KEY}
20 datasets:
21   - 2
22   - 7
```

Die Evaluierungskonfiguration umfasst die folgenden Bausteine:

- `defaultEvaluationEndpoint` ist der Standardendpunkt für die Klassifizierung. Er wird verwendet, wenn für ein Modell kein eigener Endpunkt angegeben ist. Der Endpunkt muss die in Kapitel 4.4 beschriebene API-Spezifikation erfüllen und kann relativ (gegen die Basis-URL des Evaluationsframeworks) oder absolut (für einen externen Dienst) angegeben werden.
- `maxConcurrent` gibt die maximale Anzahl parallel auszuführender Testfälle an. So lassen sich beispielsweise Rate-Limits von LLMs einhalten.

- `seed` legt einen Startwert (Seed) für reproduzierbare Evaluationsläufe fest. Er wird bei jedem Modell an die `llmProps` weitergereicht und bei der Kommunikation mit den LLMs verwendet, sofern diese einen Seed unterstützen.
- `models` enthält die zu evaluierenden Modelle. Jedes Modell besitzt ein `label` zur Identifikation und optional spezifische `llmProps`, um die Eigenschaften des verwendeten LLMs zu definieren.
- `datasets` ist eine Liste von Datensatz-ids, die in der Evaluierung verwendet werden. Die ids referenzieren die Datensätze, die in der Datenbank verwaltet werden und jeweils eine Menge von Testfällen beinhalten.

Wie im Schema in Listing A.4 gezeigt, kann jedem Modell optional ein eigener `evaluationEndpoint` zugewiesen werden, der den in `defaultEvaluationEndpoint` definierten Standard überschreibt. Dadurch lassen sich unterschiedliche Klassifizierungsalgorithmen oder -versionen gezielt pro Modell vergleichen. Ist kein spezifischer Endpunkt angegeben, greift automatisch der Standardendpunkt.

API-Keys in den `llmProps` können optional als Umgebungsvariablen referenziert werden, wie im Beispiel in Listing 6.1 gezeigt. So lassen sich sensible Daten sicher handhaben, ohne sie direkt in der Konfigurationsdatei zu speichern. Die Umgebungsvariablen werden zur Laufzeit aufgelöst und müssen daher im Kontext der Anwendung verfügbar sein.

6.4 Architektur und Komponenten

Das Evaluationsframework ist modular aufgebaut und nutzt eine Pipeline-Architektur, um eine flexible und skalierbare Evaluierung zu ermöglichen, wie es in FA02 gefordert ist. Die Architektur ist in Abbildung 6.1 dargestellt. Sie besteht aus mehreren Hauptkomponenten, die jeweils eine klar definierte Aufgabe erfüllen. Im Folgenden werden die Komponenten und ihr Zusammenspiel beschrieben.

Einstiegspunkte

Das Framework bietet zwei Einstiegspunkte zur Ausführung einer Evaluierung:

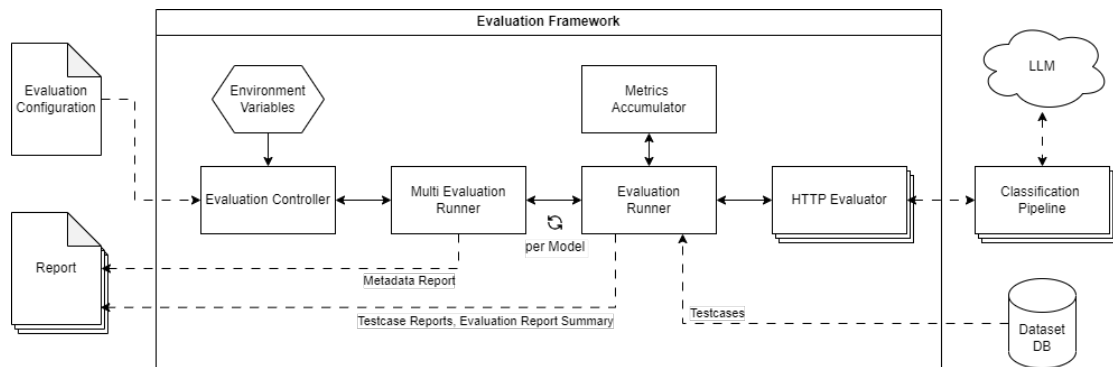


Abbildung 6.1: Architektur des Evaluationsframeworks

- **EvaluationController** als HTTP-Controller stellt REST-Endpunkte bereit, über die Evaluierungen gestartet werden können. Er akzeptiert eine YAML-Konfiguration und gibt die Ergebnisse entweder auf einmal als Markdown-Bericht oder als JSON-Stream zurück. Der Controller ermöglicht die Nutzung des Frameworks über die Weboberfläche, die in Kapitel 6.6 gezeigt wird, sowie über HTTP-APIs. Durch das Streamen der Ergebnisse können bereits abgeschlossene Testfälle sofort angezeigt werden, ohne auf das Ende der gesamten Evaluierung warten zu müssen.
- **EvaluationCommand** ist ein CLI-Befehl, der die Ausführung von Evaluierungen über die Kommandozeile erlaubt. Er liest eine YAML-Konfigurationsdatei ein, führt die Evaluierung aus und schreibt die Ergebnisse in eine Markdown-Datei. Dies eignet sich besonders für automatisierte Ausführungen, Continuous Integration oder die lokale Entwicklung.

Beide Einstiegspunkte akzeptieren die Konfiguration aus Kapitel 6.3, lösen ggf. Umgebungsvariablen auf und delegieren die Ausführung der Evaluation an den `MultiEvaluationRunner`.

Orchestrierung mit `MultiEvaluationRunner`

Der `MultiEvaluationRunner` ist für die Orchestrierung der gesamten Evaluierung verantwortlich. Er verarbeitet die Konfiguration, die mehrere Modelle und Datensätze beschreibt, und koordiniert die sequenzielle Evaluierung aller konfigurierten Modelle. Für jedes Modell ruft der `MultiEvaluationRunner` den `EvaluationRunner`

auf und übergibt diesem alle Informationen zur Ausführung der Evaluation eines einzelnen Modells.

Der `MultiEvaluationRunner` stellt zudem sicher, dass alle Modelle denselben Seed verwenden, um reproduzierbare Ergebnisse zu gewährleisten. Falls in der Konfiguration kein Seed angegeben wurde, wird an dieser Stelle einer erzeugt. Im Anschluss werden die Metadaten über die verwendeten Datensätze, Modelle, Endpunkte und den Seed gesammelt und als `MetadataReport` als Teil des Evaluationsberichts zurückgegeben. Die Teile des Berichts werden als Flow von Berichtartefakten zurückgegeben, wodurch eine streambasierte Verarbeitung ermöglicht wird. Welche Arten von Berichtartefakten es gibt, wird in Kapitel 6.5 beschrieben.

Ausführung mit `EvaluationRunner`

Der `EvaluationRunner` führt die Evaluierung für ein einzelnes Modell durch. Er lädt die Testfälle der angegebenen Testdatensätze aus der Datenbank, führt die Klassifizierung für jeden Testfall aus und sammelt die Ergebnisse. Die Testfälle werden parallel verarbeitet, wobei die Anzahl der gleichzeitigen Ausführungen durch den Parameter `maxConcurrent` in der Konfiguration gesteuert wird. Dies ermöglicht es, Rate-Limits von LLMs-Diensten einzuhalten und die Auslastung der Ressourcen zu kontrollieren.

Für jeden Testfall delegiert der `EvaluationRunner` die eigentliche Klassifizierung an den `HttpEvaluator`. Anschließend vergleicht er die erwarteten mit den tatsächlichen Ergebnissen und berechnet die Klassifikationsmetriken wie TP, FP, FN und TN. Die Ergebnisse werden in `TestCaseReport`-Objekten zusammengefasst, als Teilergebnis zurückgegeben, und an den `MetricsAccumulator` weitergeleitet.

Der `EvaluationRunner` gibt die Ergebnisse ebenfalls als Flow zurück, wodurch eine frühzeitige Rückgabe von Teilergebnissen ermöglicht wird. Dies ist besonders vorteilhaft für die Live-Ansicht in der Weboberfläche, da Testfallergebnisse sofort nach ihrer Fertigstellung angezeigt werden können.

Klassifizierung mit `HttpEvaluator`

Der `HttpEvaluator` ist für die Kommunikation mit dem Klassifizierungsendpunkt verantwortlich, der das in Kapitel 4.4 beschriebene Interface implementiert. Er nimmt das BPMN-Modell aus dem aktuellen Testfall und die `LlmProps` von dem aktuellen Modell aus der Konfiguration entgegen, baut einen HTTP-Request auf und sendet diesen an den konfigurierten Endpunkt. Nach erfolgreicher Klassifizierung extrahiert er die Liste der als kritisch identifizierten Aktivitäten aus der Antwort und gibt diese an den `EvaluationRunner` zurück.

Akkumulierung mit `MetricsAccumulator`

Der `MetricsAccumulator` sammelt die Metriken aller Testfälle eines Modells und berechnet daraus aggregierte Werte. Er ist thread-sicher implementiert und kann gleichzeitig von mehreren parallelen Evaluierungen genutzt werden. Das ist wichtig, da der `EvaluationRunner` die Testfälle parallel ausführt und somit mehrere Threads gleichzeitig auf den `MetricsAccumulator` zugreifen können.

Nach Abschluss aller Testfälle erzeugt der `MetricsAccumulator` ein `EvaluationReportSummary`-Objekt, das alle Metriken für die Evaluation eines Modells über mehrere Testfälle hinweg enthält.

Zusammenfassung

Die Architektur trennt Zuständigkeiten strikt: `MultiEvaluationRunner` koordiniert Modellläufe, `EvaluationRunner` verarbeitet Testfälle und sammelt Metriken, `HttpEvaluator` kommuniziert mit der Klassifizierungs-Pipeline, `MetricsAccumulator` aggregiert Ergebnisse pro Modell über mehrere Testfälle.

6.5 Evaluationsergebnisse

Im vorherigen Abschnitt wurde erwähnt, dass die Komponenten des Evaluationsframeworks Berichtartefakte zurückgeben. Diese sind im im Architekturbild 6.1 als

Beschriftungen über den gestrichelten Pfeilen dargestellt. Im Folgenden werden die Berichtartefakte beschrieben:

MetadataReport Berichtsartefakt, das der `MultiEvaluationRunner` zu Beginn der Evaluierung erzeugt. Es enthält Metadaten zur Evaluierung, z. B. Informationen über die Testdatensätze, die Anzahl der Testfälle sowie den verwendeten Seed. Das `MetadataReport`-Artefakt wird zuerst zurückgegeben, damit die Weboberfläche bereits Metadaten anzeigen kann, während die Evaluierung noch läuft.

TestCaseReport Berichtsartefakt, das der `EvaluationRunner` für jeden abgeschlossenen Testfall erzeugt. Es enthält u. a. die Testfall-id, das Klassifizierungsergebnis und die für diesen Testfall berechneten Metriken. `TestCaseReport`-Artefakte werden fortlaufend bereitgestellt, sobald ein Testfall abgeschlossen ist, sodass die Weboberfläche Ergebnisse unmittelbar anzeigen kann.

EvaluationReportSummary Berichtsartefakt, das der `MetricsAccumulator` am Ende der Evaluierung eines Modells erzeugt. Es fasst die aggregierten Metriken, wie z. B. *Precision*, *Recall*, *F1-Score* und *Accuracy*, sowie die Konfusionsmatrix zusammen. Das `EvaluationReportSummary`-Artefakt wird als letztes Berichtsartefakt pro Modell zurückgegeben und dient dem Modellvergleich in der Weboberfläche.

Die Informationen dieser Berichtsartefakte ermöglichen die Generierung eines ausführlichen Evaluierungsberichts, wie in FA04 gefordert. Im Folgenden ist dargestellt, welche Informationen nach Abschluss einer Evaluierung vorliegen.

Pro Testfall und Modell

Für jeden Testfall eines Modells liegen vor: die von der Klassifizierungs-Pipeline zurückgegebenen klassifizierten Aktivitäten (mit optionalen Begründungen), die gelabelten erwarteten Aktivitäten, die Zählwerte für *TP*, *FP*, *FN* und *TN* sowie eine Bild-URL zur Visualisierung des BPMN-Modells mit hervorgehobenen Aktivitäten. Aus diesen Informationen lässt sich ableiten, ob der Testfall erfolgreich war. Ein Testfall gilt als erfolgreich, wenn die klassifizierten Aktivitäten exakt den erwarteten Aktivitäten entsprechen. Technische Probleme, die während der Klassifizierung

auftreten, werden ebenfalls erfasst, z. B. Parsing-Fehler, ungültiges BPMN, Token-Limit-Überschreitungen oder Zeitüberschreitungen.

Pro Modell über alle Testfälle

Auf Modellebene stehen die Gesamtergebnisse über alle Testfälle zur Verfügung. Dazu gehören die aggregierten Kennzahlen *Precision*, *Accuracy*, *Recall* und *F1-Score* sowie eine Konfusionsmatrix mit den Gesamtwerten für *TP*, *FP*, *FN* und *TN*. Zusätzlich sind die Anzahlen der korrekt bzw. falsch klassifizierten sowie der technisch fehlgeschlagenen Testfälle aufgeführt.

Über alle Modelle

Abschließend sind die Metadaten der gesamten Evaluierung verfügbar: die verwendeten Testdatensätze, die Anzahl der Testfälle, die konfigurierten Modelle, der für die Reproduzierbarkeit verwendete Seed sowie ein Zeitstempel der Evaluierung. Zum unmittelbaren Vergleich werden die aggregierten Kennzahlen aller Modelle nebeneinander dargestellt.

6.6 Frontend

Das Frontend des Evaluationsframeworks setzt die Anforderungen FA05 und FA06 um. Es unterstützt die interaktive Konfiguration von Evaluierungen, die Live-Verfolgung des Fortschritts sowie die detaillierte Analyse der Ergebnisse bis auf Ebene einzelner Testfälle. Die Oberfläche ist so gestaltet, dass zentrale Kennzahlen wie *Accuracy*, *Precision*, *Recall* und *F1-Score*, die Konfusionsmatrix mit *TP*, *FP*, *TN*, *FN* sowie die Bestehensraten aller Modelle zunächst auf einen Blick erfasst und anschließend schrittweise vertieft werden können.

Evaluation Config
Multiple Models

Import YAML Config Download YAML Config

Default Settings

Default Evaluation Endpoint
Use preset

Preset Endpoint
Preprocessing & Prompt Engineering Analysis

Max Concurrent LLM Requests
10

Seed
Optional seed for reproducibility

Warning: Not all models support a seed, but it will be used for models that support them.

Datasets

Select Datasets
Uni Reale Scenarien Kleine Scenarien

Models Configuration

Model 1 Effective endpoint: /gdpr/analysis/prompt-engineering

Label
Deepseek Chat v3.1

Endpoint
Use default

LLM Base URL
https://openrouter.ai/api/v1

LLM Model Name
deepseek/deepseek-chat-v3.1

LLM Response Timeout (seconds)
240

API Key
\${OPEN_ROUTER_API_KEY}

Model 2 Effective endpoint: /gdpr/analysis/prompt-engineering

Label
Mistral Medium 3.1

Endpoint
Use default

LLM Base URL
https://openrouter.ai/api/v1

LLM Model Name
mistralai/mistral-medium-3.1

LLM Response Timeout (seconds)
240

API Key
\${OPEN_ROUTER_API_KEY}

Download Markdown Report Download JSON Report Upload JSON Report Start Evaluation

Abbildung 6.2: Formular zur Konfiguration einer Evaluation.

Konfigurationsansicht

Abbildung 6.2 zeigt das Formular zur Konfiguration einer Evaluierung. Sämtliche Parameter, die bereits aus der YAML-Konfiguration in Kapitel 6.3 bekannt sind, lassen sich hier setzen. Verfügbare Standardwerte, zum Beispiel der Endpunkt der in dieser Arbeit verwendeten Klassifizierungspipeline oder die in der Datenbank verfügbaren Datensätze, werden automatisch geladen.

YAML-Konfigurationen können importiert und exportiert werden, um sie zu speichern oder weiterzugeben. Unter dem Formular befinden sich Schaltflächen zum Starten der Evaluierung sowie zum Import und Export von JSON-Berichten. Auf diese Weise lassen sich Ergebnisse archivieren und später erneut laden, ohne die Evaluierung erneut ausführen zu müssen.

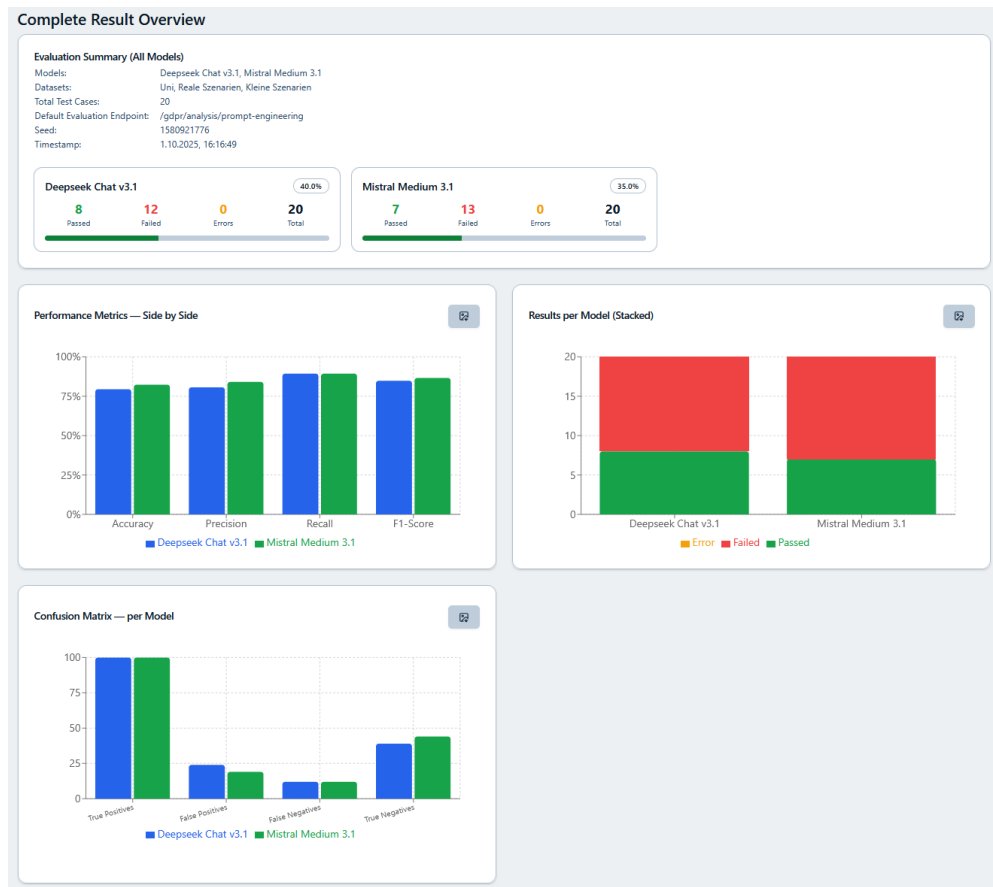


Abbildung 6.3: Gesamtübersicht einer Evaluierung mit Side-by-Side-Diagrammen.

Gesamtübersicht

Nach dem Start der Evaluierung werden die Ergebnisse pro Modell inkrementell vom Backend übermittelt und — wie in Abbildung 6.3 — angezeigt. Dadurch können Teilergebnisse bereits untersucht werden, während die Evaluierung noch läuft. Die Gesamtübersicht bietet eine kompakte Zusammenfassung pro Modell mit den Kategorien Bestanden, Nicht bestanden und Fehler sowie Side-by-Side-Diagramme aller Metriken mit allen Modellen. So lassen sich die Modelle unmittelbar nebeneinander vergleichen. Oberhalb sind die Metadaten der Evaluierung aufgeführt.

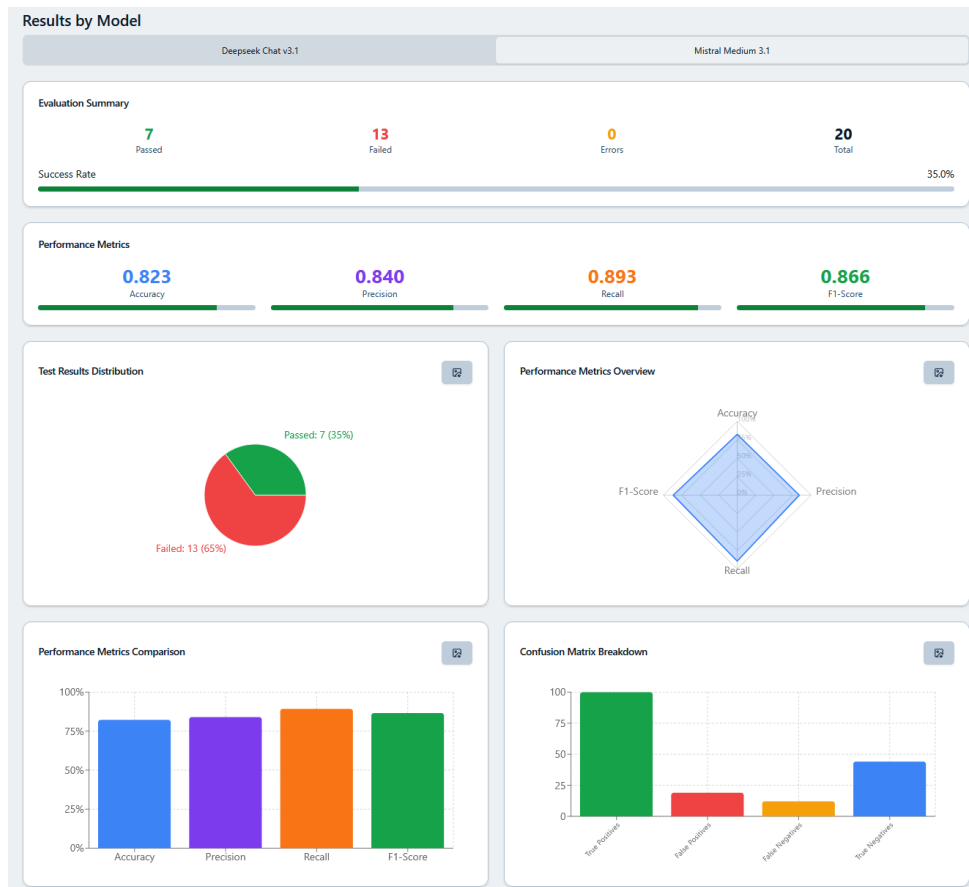


Abbildung 6.4: Modell-Detailansicht mit exemplarischen Ergebnissen.

Ergebnisse pro Modell

Für eine vertiefte Analyse stellt das Frontend für jedes Modell eine Detailansicht bereit, die alle Kennzahlen über sämtliche Testfälle aggregiert. Abbildung 6.4 zeigt diese Ansicht. Über Tabs kann zwischen den Modellen gewechselt werden, was einen schnellen Vergleich ermöglicht.

Ergebnisse pro Testfall

Neben den aggregierten Ergebnissen pro Modell lassen sich auch die Resultate einzelner Testfälle je Modell untersuchen. Abbildung 6.5 zeigt die Detailseite eines Testfalls. Sie enthält unter anderem den Status, die erwarteten gelabelten Aktivitäten und die vom Modell detektierten Aktivitäten. Zusätzlich visualisiert eine BPMN-

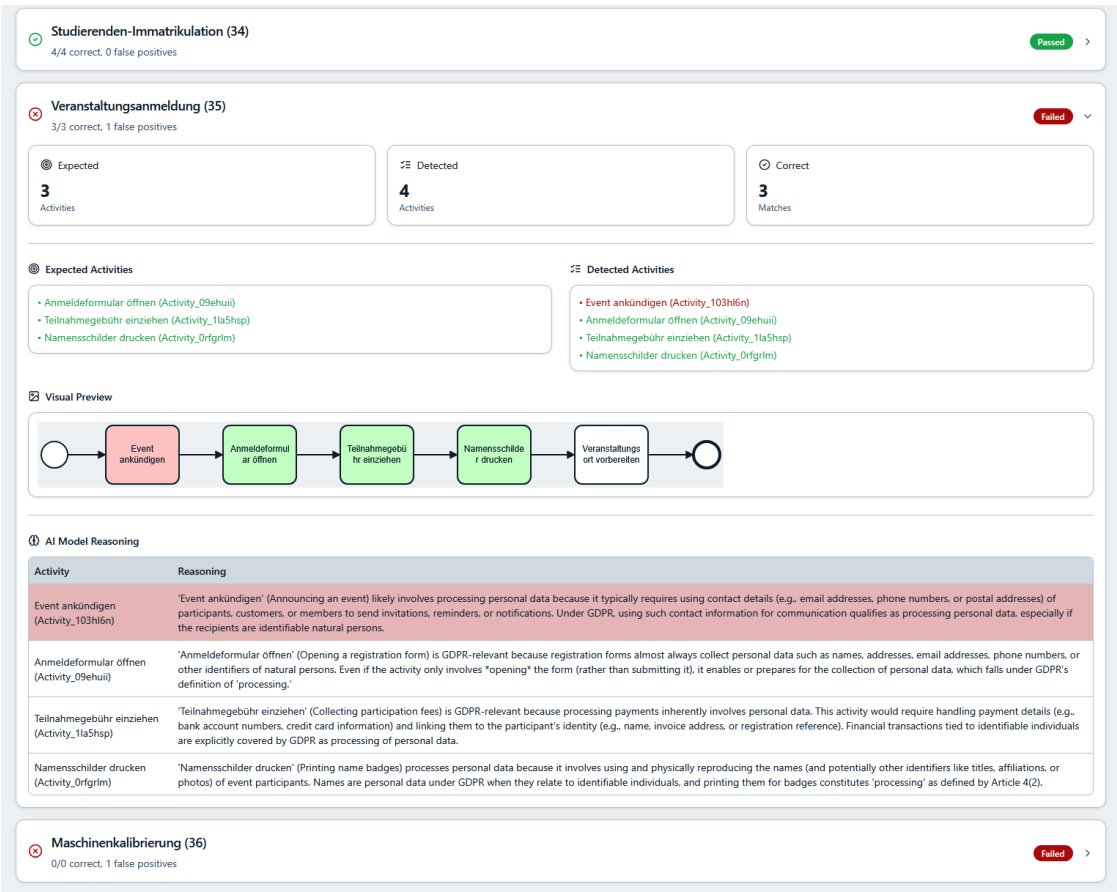


Abbildung 6.5: Detailseite eines Testfalls mit exemplarischen Ergebnissen.

Darstellung den Prozess, und die Aktivitäten sind je nach korrekter oder inkorrekt Klassifizierung farblich markiert. Falls vorhanden, wird außerdem die vom LLM gelieferte Begründung pro Aktivität angezeigt.

Abweichungen werden dadurch unmittelbar sichtbar, und typische Fehlmuster wie systematische FP bei bestimmten Aktivitätstypen lassen sich schnell erkennen. Testfälle, die aufgrund technischer Fehler nicht klassifiziert werden konnten, werden mit der entsprechenden Fehlermeldung aufgeführt.

6.7 Erweiterbarkeit

Das Evaluationsframework ist auf die Entkopplung von *Modell*, *Klassifizierungspipeline* und *Testdaten* ausgelegt. Neue Modelle werden rein konfigurationsbasiert eingebunden, indem Endpunkt, Modellname und API-Key hinterlegt werden. Dadurch ist ein Austausch ohne Codeänderungen möglich. Klassifizierungsalgorithmen bzw. Pipelines werden über einen konfigurierbaren HTTP-Endpoint ergänzt, sofern sie das in Abschnitt 4.4 definierte Interface implementieren. Dadurch lassen sich verschiedene Varianten von Algorithmen unkompliziert unter identischen Rahmenbedingungen vergleichen. Die Testdatensätze werden zur Laufzeit aus der Datenbank geladen und pro Evaluierung können unterschiedliche Datensätze ausgewählt werden. Neue Datensätze lassen sich jederzeit hinzufügen, um weitere Domänen oder Schwierigkeitsgrade abzudecken. Diese Architektur ermöglicht es, die Umgebung schrittweise zu erweitern und aktuelle Modelle, Verfahren konsistent und reproduzierbar zu evaluieren.

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

Hier kann ich auch erwähnen, dass ich aufgrund von Hardware Limitierungen Openrouter benutze, um auch auch größere Modelle zu testen. Ich wollte in meinen Ergebnissen nicht durch fehlende Hardware limitiert sein.

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 verglichen
- 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

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 herausuchen 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-Rate 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 Vorsecreening, 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 verglichen 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 fuer 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 fuer 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     /* Andere Methoden dieser Klasse sind weggelassen */
21 }

```

Listing A.3: Kern der id-Validierung und -Vervollständigung

```

1 fun resolveActivityIds(actualBpmnElements: Set<BpmnElement>):
    ↳ BpmnAnalysisResult {
2     val existingActivityIds = actualBpmnElements
3       .filter { it.type.lowercase().contains("task") }
4       .map { it.id }.toSet()
5
6     val resolvedDistinct = elements.mapNotNull { element ->
7       val resolvedId = resolveActivityIdUniquely(element.id,
8     ↳ existingActivityIds)
9       resolvedId?.let { if (it == element.id) element else element.
10     ↳ copy(id = it) }
11     }.distinctBy { it.id }
12
13     return BpmnAnalysisResult(elements = resolvedDistinct)
14 }
15
16 private fun resolveActivityIdUniquely(partialId: String,
17   ↳ existingActivityIds: Set<String>): String? {
18     if (partialId in existingActivityIds) return partialId
19     existingActivityIds.filter { it.startsWith(partialId) }.
20     ↳ singleOrNull()?.let { return it }
21     return existingActivityIds.filter { it.contains(partialId) }.
22     ↳ singleOrNull()
23 }

```

Listing A.4: Schema der YAML-Evaluationskonfiguration

```

1 {
2     "$schema": "https://json-schema.org/draft/2020-12/schema",
3     "$ref": "#/definitions/Configuration",
4     "definitions": {
5         "Configuration": {
6             "type": "object",

```

```

7      "additionalProperties": false,
8      "properties": {
9          "defaultEvaluationEndpoint": {
10             "type": "string"
11         },
12         "maxConcurrent": { "type": "integer" },
13         "models": {
14             "type": "array",
15             "items": { "$ref": "#/definitions/Model" }
16         },
17         "datasets": {
18             "type": "array",
19             "items": { "type": "integer" }
20         },
21         "seed": { "type": "integer" }
22     },
23     "required": [
24         "defaultEvaluationEndpoint",
25         "models",
26         "datasets",
27     ],
28     "title": "Configuration"
29 },
30 "Model": {
31     "type": "object",
32     "additionalProperties": false,
33     "properties": {
34         "label": { "type": "string" },
35         "evaluationEndpoint": { "type": "string" },
36         "llmProps": { "$ref": "#/definitions/LlmProps" }
37     },
38     "required": [ "label" ],
39     "title": "Model"
40 },
41 "LlmProps": {
42     "type": "object",
43     "additionalProperties": false,

```

```

44         "properties": {
45             "baseUrl": {
46                 "type": "string",
47                 "format": "uri",
48                 "qt-uri-protocols": [ "https" ]
49             },
50             "modelName": { "type": "string" },
51             "apiKey": { "type": "string"},
52             "timeoutSeconds": { "type": "number" }
53         },
54         "required": [],
55         "title": "LlmProps"
56     }
57 }
58 }

```

Listing A.5: Zusammengefasster Logauszug zum Retry-Mechanismus

```

1 2025-10-03T19:11:51.152+02:00 INFO BpmnExtractor : Extracting BPMN
   ↳ elements from XML
2
3 # 1) Erste Anfrage an das LLM (gekuerzt: Prompt/Headers/Body)
4 2025-10-03T19:11:51.156+02:00 INFO LoggingHttpClient : HTTP POST
   ↳ https://openrouter.ai/api/v1/chat/completions
   model: openai/gpt-oss-20b
   messages: [system: (System-Prompt), user: (User-Prompt mit
   ↳ BpmnElement-Liste und Format-Anweisung)]
5
6
7
8 # 2) Antwort des LLM mit fehlerhaftem JSON (verkuerzt)
9 2025-10-03T19:11:56.671+02:00 INFO LoggingHttpClient : HTTP 200
10 assistant:
11 {
12     "elements": [
13         { "id": "Activity_09ehuii", "reason": "...", "isRelevant": true },
14         { "id": "Activity_1la5hsp", "reason": "...", "isRelevant": }
15         ↳ <-- fehlender Bool-Wert
16         { "id": "Activity_0rfgrlm", "reason": "...", "isRelevant": true }
17     ]

```

```
17 }
18
19 # 3) Parser-Fehler + Retry-Ankuendigung (gekuerzt)
20 2025-10-03T19:11:56.691+02:00 WARN SafetyNet : Parsing failed.
    ↳ Attempting to fix JSON and retry... (Attempt 1 of 2)
21 dev.langchain4j.service.output.OutputParsingException:
22   Caused by: com.fasterxml.jackson.core.JsonParseException:
23   Unexpected character ('}') ... at elements[1].isRelevant
24
25 # 4) Zweite Anfrage zum beheben des JSON mit Chat-Verlauf und
    ↳ Fehlermeldung (n-mal wiederholt, bis erfolgreich)
26 2025-10-03T19:11:56.721+02:00 INFO LoggingHttpClient : HTTP POST
    ↳ https://openrouter.ai/api/v1/chat/completions
27 messages: [
28   system: (System-Prompt),
29   user: (User-Prompt mit BpmnElement-Liste und Format-Anweisung),
30   assistant: (Fehlerhafte JSON-Antwort),
31   system: (Fix-JSON System-Prompt),
32   user: (Fehlermeldung)
33 ]
34
35 # 5) Korrigierte JSON-Antwort des LLM
36 2025-10-03T19:12:01.519+02:00 INFO LoggingHttpClient : HTTP 200
37 assistant:
38 {
39   "elements": [
40     { "id": "Activity_09ehuii", "reason": "...", "isRelevant": true },
41     { "id": "Activity_1la5hsp", "reason": "...", "isRelevant": true },
    ↳ <-- jetzt mit Bool-Wert
42     { "id": "Activity_0rfgrlm", "reason": "...", "isRelevant": true }
43   ]
44 }
45
46 # 6) Erfolgreiches Parsing und Weiterverarbeitung
47 2025-10-03T19:12:01.519+02:00 INFO PromptBpmnAnalyzer : BPMN
    ↳ Analysis Result: elements=[... isRelevant=true ...]
```

Abbildungsverzeichnis

2.1	Die relevanten BPMN Elemente in Beziehungen zueinander	8
2.2	Beispiel einer Datenassoziation als Datenschutzsinal.	9
4.1	BPMN-Diagramm der Klassifizierungspipeline.	21
4.2	Sandbox im Frontend mit hervorgehobenen kritischen Aktivitäten nach Analyse.	32
4.3	Begründung der Klassifikation durch das LLM in der Sandbox.	33
5.1	Labeling-Editor im Labeling-Modus mit exemplarischem Modell.	35
5.2	Übersicht der Datensätze im Labeling-Tool.	36
6.1	Architektur des Evaluationsframeworks	45
6.2	Formular zur Konfiguration einer Evaluation.	50
6.3	Gesamtübersicht einer Evaluierung mit Side-by-Side-Diagrammen.	51
6.4	Modell-Detailansicht mit exemplarischen Ergebnissen.	52
6.5	Detailseite eines Testfalls mit exemplarischen Ergebnissen.	53

Listings

4.1	Interne BPMN-Repräsentation je Flow-Element.	21
4.2	Deklarative Schnittstelle für die Analyse eines BPMN-Prozesses. . .	24
4.3	JSON-Schema der <code>llmProps</code>	29
4.4	JSON-Schema der API-Antwort.	30
4.5	Beispielaufruf des Klassifizierungsendpunkts mit <code>curl</code>	31
6.1	Beispiel einer Evaluierungskonfiguration in YAML.	43
A.1	System-Prompt fuer die DSGVO-Klassifikation von BPMN-Aktivitäten	65
A.2	Antworttyp fuer die Klassifizierung	68
A.3	Kern der <code>id</code> -Validierung und -Vervollständigung	69
A.4	Schema der YAML-Evaluationskonfiguration	69
A.5	Zusammengefasster Logauszug zum Retry-Mechanismus	71

Tabellenverzeichnis

5.1	Eckdaten der verwendeten Datensätze	37
5.2	Beispielhafte Aktivitäten und Label	38

Literatur

- [1] European Data Protection Board (EDPB). *1.2 billion euro fine for Facebook as a result of EDPB binding decision*. Mai 2023. URL: https://www.edpb.europa.eu/news/news/2023/12-billion-euro-fine-facebook-result-edpb-binding-decision_en (besucht am 02.10.2025).
- [2] DeepSeek AI. *DeepSeek AI Open Source Hugging Face Models*. 2025. URL: <https://huggingface.co/deepseek-ai> (besucht am 17.07.2025).
- [3] Mistral AI. *Mistral AI*. 2025. URL: <https://mistral.ai/> (besucht am 21.09.2025).
- [4] Mistral AI. *Mistral AI - Structured Output*. 2025. URL: https://docs.mistral.ai/capabilities/structured-output/structured_output_overview/ (besucht am 11.07.2025).
- [5] 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).
- [6] Harrison Blake und Dorcas Esther. „Impact of Dataset Diversity on Model Evaluation Metrics“. In: (Jan. 2025). URL: https://www.researchgate.net/publication/387898702_Impact_of_Dataset_Diversity_on_Model_Evaluation_Metrics.
- [7] 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.
- [8] Bundesministerium der Justiz. *Gesetz über außergerichtliche Rechtsdienstleistungen (Rechtsdienstleistungsgesetz - RDG)*. <https://www.gesetze-im-internet.de/rdg/>. Dez. 2007. (Besucht am 15.08.2025).

- [9] 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).
- [10] 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).
- [11] Antonio Capodieci u. a. „BPMN-Enabled Data Protection and GDPR Compliance“. In: *IS-EUD Workshops*. 2023. URL: <https://api.semanticscholar.org/CorpusID:259099646>.
- [12] 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>.
- [13] Datenschutzticker. *Gericht bestätigt Rekordbußgeld gegen Amazon*. Apr. 2025. URL: <https://datenschutzticker.de/2025/04/gericht-bestaetigt-rekordbussgeld-gegen-amazon/> (besucht am 02.10.2025).
- [14] 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.
- [15] 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.
- [16] Camunda Services GmbH. *bpmn-js - BPMN 2.0 viewer and editor*. 2025. URL: <https://bpmn.io/toolkit/bpmn-js/> (besucht am 20.06.2025).
- [17] Camunda Services GmbH. *BPMN.io - Web-based tooling for BPMN, DMN and Forms*. 2025. URL: <https://bpmn.io/> (besucht am 22.09.2025).
- [18] Camunda Services GmbH. *Camunda Platform*. 2025. URL: <https://camunda.com/de/> (besucht am 22.09.2025).

- [19] Ashish Hooda u. a. *PolicyLR: A Logic Representation For Privacy Policies*. 2024. arXiv: 2408.14830 [cs.CR]. URL: <https://arxiv.org/abs/2408.14830>.
- [20] 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>.
- [21] Adam Tauman Kalai u. a. *Why Language Models Hallucinate*. 2025. arXiv: 2509.04664 [cs.CL]. URL: <https://arxiv.org/abs/2509.04664>.
- [22] 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).
- [23] Langchain4j. *LangChain4j Documentation 2025*. 2025. URL: <https://docs.langchain4j.dev/> (besucht am 14.06.2025).
- [24] 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>.
- [25] Christopher D. Manning, Prabhakar Raghavan und Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008. ISBN: 9780521865715. URL: <https://nlp.stanford.edu/IR-book/>.
- [26] Shervin Minaee u. a. *Large Language Models: A Survey*. 2025. arXiv: 2402.06196 [cs.CL]. URL: <https://arxiv.org/abs/2402.06196>.
- [27] 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>.
- [28] Joshua Noble. *What is LLM Temperature?* URL: <https://www.ibm.com/think/topics/llm-temperature>.
- [29] 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).

- [30] OpenAI. *Function calling and other API updates*. <https://openai.com/index/function-calling-and-other-api-updates/>. 2023. (Besucht am 10.07.2025).
- [31] OpenAI. *Hello GPT-4o*. Mai 2024. URL: <https://openai.com/index/hello-gpt-4o/> (besucht am 21.07.2025).
- [32] OpenAI. *Model Overview*. 2025. URL: <https://platform.openai.com/docs/models> (besucht am 18.09.2025).
- [33] OpenAI. *OpenAI - Structured model outputs*. URL: https://docs.mistral.ai/capabilities/structured-output/structured_output_overview/ (besucht am 11.07.2025).
- [34] 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>.
- [35] Quarkiverse Contributors. *AI Services Reference (Quarkus LangChain4j)*. 2025. URL: <https://docs.quarkiverse.io/quarkus-langchain4j/dev/ai-services.html> (besucht am 14.06.2025).
- [36] Alibaba Qwen. *Qwen Open Source Hugging Face Models*. 2025. URL: <https://huggingface.co/Qwen> (besucht am 17.07.2025).
- [37] 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/>.
- [38] Reuters. *Amazon hit with record EU data privacy fine*. Juli 2021. URL: https://www.reuters.com/business/retail-consumer/amazon-hit-with-886-million-eu-data-privacy-fine-2021-07-30/?utm_source=chatgpt.com (besucht am 02.10.2025).

- [39] 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>.
- [40] 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>.
- [41] 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>.
- [42] Marina Sokolova und Guy Lapalme. „A systematic analysis of performance measures for classification tasks“. In: *Information processing & management* 45.4 (2009), S. 427–437. URL: <https://www.sciencedirect.com/science/article/pii/S0306457309000259>.
- [43] 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>.
- [44] 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