universität
uulm

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

Identifying GDPR-Critical Tasks in Business Processes using 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 15. September 2025

Satz: PDF-LaTeX 2ε

Inhaltsverzeichnis

Abkürzungen

1 Einleitung

1.1 Motivation

- Businessprozesse sind überall.
- Datenschutz relevant in Europa.
- Datenschutzprüfungen in Prozessen sind kosten- und zeitintensiv.
- Fehlerhafte Unterentdeckung ist ggf. kritisch (False Negatives).
- LLMs versprechen schnelles Screening (vor allem europäische Open-Source-Modelle sind interessant); dafür werden jedoch noch belastbare Evidenz und reproduzierbare Vergleiche benötigt.

1.2 Problemstellung

- Es fehlt ein standardisierter, reproduzierbarer Vergleich verschiedener Modelle für die Aufgabe, Aktivitäten in Business-Prozessen nach *kritisch* oder *nicht kritisch* zu klassifizieren.
- Besonders interessant ist, wie sich Open-Source-EU-Modelle schlagen und welche Trade-offs ggf. entstehen.
- Warum ist das schwer -> BPMN-Modelle sind multimodal (Text + Struktur), Datenobjekte fehlen oft in der Modellierung -> Datennutzung muss impliziert werden.

1.3 Zielsetzung und Beiträge

- Eine Klassifizierungspipeline für Businessprozesse (Aktivitäten nach datenschutzkritisch/nicht kritisch klassifizieren, binäre Klassifikation).
- Ein Evaluationsframework zum Vergleich beliebiger Modelle und Algorithmen über eine einheitliche Schnittstelle unter der Nutzung gelabelter Businessprozesse.
- Eine Labelingsoftware für das Erstellen und Labeln von Datensätzen für die Evaluierung.
- Erstellung eines repräsentativen Datensatzes aus gelabelten Businessprozessen (inkl. Erklärung, nach welchen Kriterien gelabelt wurde).
- Empirische Befunde, um die Forschungsfragen beantworten zu können (bei Seed: Befunde nachprüfbar machen, Code + Config + Seeds bereitstellen).

1.4 Forschungsfrage und Unterfragen

- **Forschungsfrage:** Wie zuverlässig identifizieren große Sprachmodelle DSGVO-kritische Aktivitäten in BPMN-Prozessmodellen?
- **Unterfragen:** Wie gut schneiden die EU-Modelle gegen internationale ab? Wie gut schneiden große Modelle gegen kleine ab? Welche Open-Source-Modelle (insbesondere aus Europa) schneiden besonders gut ab? Wie gut schneiden Open-Source-Modelle gegen kommerzielle Modelle ab (z. B. GPT-4o)?
- Begründung, warum binäres Labeln zunächst hinreichend für ein erstes Screening ist. Eine Detailprüfung folgt im nachfolgenden Schritt (nicht mehr Teil dieser Masterarbeit).

1.5 Aufbau der Arbeit

- Kurzer Überblick über jedes Kapitel (vom Kontext über Algorithmus, Framework (Labeling & Evaluierung) hin zu Durchführung, Ergebnisse und abschließend Diskussion).

2 Hintergrund und verwandte Arbeiten

2.1 DSGVO

- Definitionen: Personenbezogene Daten, Verarbeitung, typische Auslöser für “kritisch” (Erhebung, Nutzung, Speichern, Übermitteln, Löschen, . . .)
- Abgrenzung, dass es sich hier bei den Klassifizierungen durch die Algorithmen nur um ein Risiko-Screening handelt und nicht um eine Rechtsberatung (Sollte ich vielleicht erwähnen, damit auch die Ergebnisse am Ende besser eingeordnet werden können)

2.2 BPMN

- Relevante Elemente wie Aktivitäten, Datenobjekte/-speicher, Nachrichtenflüsse, Pools/Lanes, Assoziationen
- Rolle von Elementen, welche zusätzliche Hinweise auf Verarbeitung von personenbezogenen Daten geben (Bspw. Aktivität holt sich Informationen aus einer Datenbank wo Kundendaten gespeichert sind; dargestellt durch Assoziation)
- BPMN-XML erläutern, weil das auch das Format ist in dem Businessprozesse gespeichert sind und welches ich auch als Eingabe für die Klassifizierungspipeline nutze

- Stabile IDs für Elemente im Businessprozess sind essenziell, damit die Ausgabe des LLM immer auf die gleichen Aktivitäten bezieht und man das Ergebnis deterministisch überprüfen kann
- Kleines Beispiel wie eine Datenassoziation signalisieren kann, dass eine Aktivität datenschutzkritisch ist, was nur mit dem Namen der Aktivität nicht möglich ist

2.3 LLMs

- Basiert auf Transformer Architektur
- Prompting (System Message, User Prompt)
- Zero-/Few-Shot (Später Interessant für Prompt des Algorithmus)
- JSON-konforme Ausgaben (Schema-Enforcing, Wichtig für die Klassifizierung)
- typische Fehlerbilder (Halluzination, ungültiges JSON) + Gegenmaßnahmen (Retry/Repair) (Im Klassifizierungsalgorithmus gibt es Reparaturmechanismen wir das herausfiltern ungültiger Aktivität IDs)
- Gpt-4o als aktuellen Standard präsentieren

2.4 Verwandte Arbeiten

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

3 Problemdefinition und Zielkriterien

3.1 Aufgabenstellung

- Eingabe ist BPMN-XML; Ausgabe ist eine Menge von Aktivitäts-IDs, die als kritisch klassifiziert worden sind. Optional auch mit Erklärung vom LLM warum es sich so entschieden hat
- Definition was als kritisch zu klassifizieren ist: Eine Aktivität ist kritisch, wenn sie personenbezogene Daten verarbeitet (Hier nochmal genauer definieren nach DSGVO), einschließlich Nutzung bereits vorhandener Daten.

3.2 Qualitätsziele

// TODO Hier prüfen, ob ich das so schon hier sagen kann oder das erst bei der Diskussion thematisiert werden soll, dass mir der hohe Recall eher positiv aufgefallen ist, weil es des öfteren plausible Erklärungen dazu gab

- Hauptziel ist ein hoher Recall mit minimalen False Negatives, bei noch akzeptabler Precision und somit auch überschaubaren False Positives

// TODO Aktuell unterstützt meine Evaluierung noch keine Seeds, aber das sollte ich noch umsetzen können und dann Temperature bei den LLMs auf 0 setzen

- Sekundäres Ziel ist Stabilität über mehrere Seeds hinweg
- Hier könnte ich noch bestimmte Wertebereiche vorschlagen, sie sinnvoll wären wie Recall $\geq 80\%$, False Positives $\leq 1{,}5/$Prozess. Unbedingt dafür noch in anderen Papern nach guten Werten schauen, wenn ich das machen sollte

3.3 Scope und Annahmen

// TODO Das muss ich noch anpassen

- Sprachen sind DE und EN
- Keine Bewertung der Rechtmäßigkeit an sich, sondern nur ein Screening nach kritischen Aktivitäten (Wie ein Vorfilter)
- Risiken und Grenzen (Bspw. schlechte Einschätzung des LLM wenn Artefakte wie Datenobjekte im BPMN-Modell fehlen) werden transparent gemacht

4 Klassifizierungsalgorithmus (Design und Implementierung)

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

4.1 Ziel und Annahmen

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

4.2 BPMN Preprocessing

- BPMN-XML wird geparst
- Relationen von Flow-Elementen zu anderen Elementen werden extrahiert:

```
data class BpmnElement(
    val type: String,
    val id: String,
    val documentation: String? = null,
    val name: String? = null,
    val outgoingFlowElementIds: List<String> = emptyList(),
    val incomingFlowElementIds: List<String> = emptyList(),
    val incomingDataFromElementIds: List<String> = emptyList(),
    val outgoingDataToElementIds: List<String> = emptyList(),
    val associatedElementIds: List<String> = emptyList()
)
```

- Dadurch entsteht für jedes Element ein strukturierter Kontext (id, name, pool, lane, eingehende Elemente, ...)
- Dieser Kontext wird später im Prompt genutzt, um dem LLM alle notwendigen Informationen strukturiert bereitzustellen

4.3 Prompt Engineering

- Referenz auf Zero-Shot/Few-Shot (?)
- Prompt-Sprache (Passendes Paper referenzieren)
- Erklärung des System-Prompt (Anleitung was als kritisch klassifiziert werden soll, Auflistung wichtiger DSGVO-kritischer Inhalte und Definitionen aus der DSGVO wie “Verarbeitung”, Definition des Ausgabeformats mit LangChain4j) (System-Prompt im Anhang beifügen oder hier direkt integrieren)
- Was steht im User-Prompt drin

4.4 Validierung der Ausgabe

- Ausgabeschema wird in LangChain4j deklarativ beschrieben
- Erklären was LangChain4j im Hintergrund tut, damit das Schema eingehalten wird und die Ausgabe das korrekte Format hat

```
data class AnalysisResponse(
    val criticalElements: List<CriticalElement>
) {
    data class CriticalElement(
        val id: String,
        val name: String?,
        val reason: String
    )
}
```

- Abgleichen der ausgegebenen IDs mit den vorhandenen aus dem Prozess. ggf. unzulässige entfernen

// TODO Entscheiden, ob ich hier noch Ablationen brauche (Falls ja hier noch ein Kapitel darüber einbauen) wie eine Baseline ohne zusätzliches Prompt Engineering, Varianten wo beim Preprocessing Lanes und Pools, Datenobjekte etc. weggelassen werden

4.5 API

- Klassifizierungspipeline ist über HTTP-Endpunkt aufrufbar, wo das BPMN-XML und ggf. Attribute zum Überschreiben des genutzten LLMs übergeben werden
- Klassifizierungspipeline kann außerdem lokal über CLI gestartet werden und legt Ergebnisse in Datei ab

- Dadurch kann die Klassifizierung leicht in das Evaluationsframework eingebunden werden, während das Evaluationsframework flexibel bleibt und beliebige Klassifizierungsalgorithmen benutzen kann, welche ebenfalls die gleiche HTTP-Schnittstelle anbieten
- Beispielaufruf des Klassifizierungsendpunkts
- Irgendwo hier oder auch in einem nächsten Abschnitt die “Schnittstelle” mit BPMN-XML rein und JSON mit Liste der kritischen Elemente (evtl. mit Begründung) wieder. Durch diese Vereinheitlichung ist es möglich verschiedene Algorithmen in dem Evaluationsframework zu vergleichen (Stichwort Standardisierung). Vielleicht gehe ich auch soweit, dass ich den Standard für zukünftige Arbeiten propose, damit spätere Arbeiten meinen Standard zum Vergleich nutzen können

4.6 Nutzung über Frontend

- Die Klassifizierung kann über eine Sandbox im Frontend genutzt werden. Dort können auch die LLM-Parameter angepasst werden, um andere Modelle testen zu können
- Sandbox ist ein voller BPMN-Editor basierend auf BPMN.js und bietet zusätzlich die Funktionalität zur Klassifizierung an
- Als kritisch klassifizierte Elemente werden im Editor farblich hervorgehoben

5 Evaluationsframework

5.1 Anforderungen und Use-Cases

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

5.2 Architektur und Komponenten

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

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

5.3 Konfiguration einer Evaluierung

- Evaluierung kann mit einer YAML Datei konfiguriert werden

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

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

5.4 Testdaten

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

5.5 Generierte Resultate

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

5.6 Visualisierung im Frontend

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

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

5.7 Erweiterbarkeit

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

6 Labeling und Datensätze

6.1 Labeling Tool

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

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

6.2 Quellen und Eigenschaften der Datensätze

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

  - Kleine Prozesse mit maximal 5 Aktivitäten und keinen weiteren Elementen

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

6.3 Labeling-Guide

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

7 Modellauswahl

7.1 Kriterien

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

7.2 Modellvorstellung

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

8 Versuchsaufbau

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

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

9 Durchführung

9.1 Experimente

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

9.2 Fehlerklassen

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

10 Ergebnisse

10.1 Gesamtübersicht

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

10.2 Analyse

- Aufschlüsselung nach Datensätzen
- Aufzeigen sichtbarer Muster

10.3 Antworten auf Forschungsfragen

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

10.4 Fallstudien

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

10.5 Robustheit

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

11 Diskussion

11.1 Interpretation der Befunde

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

11.2 Hoher Recall vs. Präzision

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

11.3 EU-Modelle

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

11.4 Open-Source Modelle

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

11.5 Modellgrößen

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

11.6 Grenzen

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

12 Zusammenfassung

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

13 Aussicht

Unter anderem das hier, evtl noch mehr:

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

A Quelltexte

In diesem Anhang sind einige wichtige Quelltexte aufgeführt.

```
1 public class Hello {
2     public static void main(String[] args) {
3         System.out.println("Hello World");
4     }
5 }
```

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