KI-gestützte Übersetzung natürlichsprachlicher Prüfungslogik in DSL-Skripte

am Beispiel von Plausibilitätsprüfungen für digitale Antragsformulare

zur Erlangung des akademischen Grades
Master of Science
vorgelegt von

Florian Lemnitzer

am
02.03.2026

Technische Hochschule Brandenburg
Fachbereich: Wirtschaft
Studiengang: Master Digitalisierung und Management
Laufendes Semester: 5. Semester
E-Mail: lemnitze@th-brandenburg.de
Matrikelnummer: 20236222
Erstgutachten: Prof. Dr. Vera Meister
Zweitgutachten: M. Sc. Niels Gundermann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

deg data experts gmbh.

DSL domänenspezifische Sprache.

GIS geografisches Informationssystem.

KI künstliche Intelligenz.

XML Extended Markup Language.

1 Einleitung

(GEPLANTE SEITENZAHL: 3 - 6 | T: 01.02.26)

Leitfragen

Was ist das konkrete Thema in einem Satz?

Warum ist das Thema relevant für Praxis und Forschung?

Welches reale Problem adressiert die Arbeit (konkret: Plausibilitätsprüfungen in digitalen Formularen)?

Für welche Zielgruppe ist die Arbeit relevant (Entwickler, Fachanwender, Behörden, Forschende)?

Welchen Beitrag liefert die Arbeit knapp zusammengefasst?

Wie ist die Struktur der Arbeit (kurzer Fahrplan der Kapitel)?

Welche Begriffe/Abkürzungen werden zentral verwendet?

2 Problemstellung

(Anmerkung Niels: Ich finde, dass es dir hier sehr gut gelungen ist, dich kurz zu halten, aber trotzdem alles anzuschneiden. Die Dichte der Informationen im Text ist sehr hoch
Gerade deswegen bietet sich hier eine bildliche Übersicht an, in der die relevanten Aspekte eingeordnet werden. Das macht es dem Leser leichter, wichtige Begriffe (wie Validierungslogik, Skripte, Formular, Formularverwalter, Live-Validierungen, etc.) einzuordnen/wiederzufinden.) (Geplante Seitenzahl: 3 - 6 | T: 16.11.25)

```
Leitfragen

Welches konkrete Praxisproblem existiert heute beim Übersetzen
natürlichsprachlicher Prüfungslogik in DSL-Skripte?

Welche Stakeholder haben welches Interesse und welche Anforderungen?

Welche Nachteile haben vorhandene Workflows?

Warum sind existierende technische Lösungen unzureichend (konkret für
KI/DSL/Forms)?

Welche Folgen hat das Problem?

Welche Annahmen liegen der Problemformulierung zugrunde?
```

Die data experts gmbh (deg)[1] entwickelt einen Webclient, über den Landwirte Fördermittel beantragen können. Dafür füllen sie Antragsformulare aus und hinterlegen Flächendaten im geografischen Informationssystem (GIS).

Der Webclient erleichtert das Ausfüllen der Formulare durch eine Live-Validierung der Antragsdaten. Er gibt Meldungen aus, wenn die Datenkonstellation bestimmte Bedingungen erfüllt. Diese Meldungen enthalten Bearbeitungshinweise oder Fehlermeldungen, die Antragsteller dabei unterstützen, inhaltlich korrekte Anträge einzureichen.

[1] Bei der Kleinschreibung des Namens handelt es sich um ein Stilmittel (vgl. data experts gmbh).

Beispiel: Wenn eine deutsche IBAN angegeben wird, dann muss diese 22 Stellen haben. Sonst wird dem Anwender folgende Fehlermeldung ausgegeben: „Die von Ihnen eingegebene IBAN hat nicht die erforderlichen 22 Stellen.“

2.1 Live-Validierungen

Die Live-Validierungen werden von den Auftraggebern definiert. Dafür verwenden sie den sogenannten Formularverwalter. Im Formularverwalter können sie Formulare gestalten und Vorgaben zur Verarbeitung der Daten machen (z. B. Einreichzeiträume, Vortragen von Daten, Schnittstellen, Validierungen und weitere).

Die Validierungslogik wird in natürlicher Sprache erfasst. Dadurch können die Auftraggeber der deg Validierungen beauftragen, ohne dass sie ein technisches Verständnis benötigen. Anschließend wird die Logik von einem Entwickler der deg in Form eines Skriptes abgebildet. Die Skripte werden in der domänenspezifischen Sprache (DSL) „formScript“ entwickelt – eine Skriptsprache der deg, die speziell zur performanten Auswertung und Manipulation von Formulardaten geschaffen wurde. Diese Skripte werden dann vom Webclient während der Bearbeitung des Antrags ausgeführt, um die Daten live zu validieren.

2.2 Durchlaufzeiten

Der Zeitraum zur Beantragung von Agrarfördermitteln geht von März bis Mai. Zwischen dem Anfang des Jahres und dem Beginn der Antragstellung nehmen die Auftraggeber viele Veränderungen an den Vorgaben vor. Im ersten Quartal des Jahres beauftragen sie durchschnittlich über 300 Mal Anpassungen an Formularen oder zugehörigen Vorgaben. Durch diese erhöhte Arbeitslast steigt die Durchlaufzeit von der Beauftragung bis zur Auslieferung.

Beispiel: Die validen Erstantragsjahre für unterschiedliche Förderprogramme ändern sich jedes Jahr. In der Validierungs-Vorgabe „EAJ_Prüfung“ und im zugehörigen Skript mussten die Jahreszahlen angepasst werden. Das ist eine kleine Anpassung, die am 04.12.2024 beauftragt und am 07.02.2025 ausgeliefert wurde.

In den Wochen vor Beginn der Antragstellung fallen lange Durchlaufzeiten negativ auf. Sie führen dazu, dass das Programm erst spät getestet werden kann. Schwerwiegende Fehler, die erst kurz vor der Antragstellung auffallen, haben in der Vergangenheit bereits dazu geführt, dass das Einreichen von Anträgen zunächst gesperrt werden musste, bis die deg eine Fehlerkorrektur ausliefern konnte.

2.3 Automatische Übernahme

Um die Verzögerungen zu reduzieren, wurde bereits für einige Arten von Vorgaben eine automatische Übernahme vom Formularverwalter in den Programmcode ermöglicht. Einreichzeiträume werden beispielsweise in Form von zwei Datums-Feldern definiert und auch bei Schnittstellenbeschreibungen wird eine technisch auswertbare Form erzwungen.

Für die Live-Validierungen wurde den Auftraggebern ein Werkzeug angeboten, mit dem sie selbst eine technisch auswertbare Repräsentation hätten erstellen können. Es hätte die grafische Programmierung von Validierungslogik ermöglicht. Dieses wurde abgelehnt, mit der Begründung, dass es für die Auftraggeber zu komplex sei, die Validierungslogik konfigurativ abzubilden. **(FRAGE AN VERA: SOLLTE ICH HIER EIN INTERNES BERATUNGSPROTOKOLL ALS QUELLE ANFÜGEN?)**

Automatische Übernahmen funktionieren nur, wenn Vorgaben technisch strukturiert vorliegen (z. B. Datumsfelder). Validierungsregeln liegen in natürlicher Sprache vor und sind fachsprachlich und inkonsistent formuliert. Hinzu kommt die proprietäre DSL mit spezifischer Semantik. Bisher existiert kein Verfahren, das die in natürlicher Sprache formulierte Validierungslogik automatisch in die

proprietäre DSL des Webclients überträgt. Ob künstliche Intelligenz (KI)-gestützte Verfahren hierfür geeignet sind, ist Gegenstand dieser Arbeit.

Von einer automatisierten Übernahme von Validierungs-Vorgaben wären folgende Interessensgruppen betroffen:

Auftraggeber Die Auftraggeber der deg wollen ihre Vorgaben weiterhin in natürlicher Sprache erfassen, da ihnen andere Repräsentationen einer Prüflogik zu komplex sind. Sie wünschen sich schnellere Durchlaufzeiten, damit sie ihre Anpassungen früh genug im Programm testen können.

Entwickler Eine automatische Übernahme von Vorgaben entlastet Entwickler. Für den Fall, dass manuelle Anpassungen notwendig sind, muss die Wartbarkeit erhalten bleiben.

Qualitätssicherung Wenn mehr Vorgaben automatisiert übernommen werden, bedeutet das eine erhöhte Last für die Qualitätssicherung, da auch die automatisch übernommenen Vorgaben getestet werden. Die Fehlerquote der automatischen Übernahme sollte deshalb gleich oder geringer als die der manuellen Entwicklung sein.

Daraus ergibt sich folgende Problemstellung:
Die Validierungslogik soll automatisch in den Webclient übernommen werden können, ohne das Erfassen der Logik für die Auftraggeber zu verkomplizieren.

Die Problemstellung basiert auf folgenden Annahmen:

1. Die natürlichsprachlichen Beschreibungen der Auftraggeber sind sprachlich uneinheitlich. Inhaltlich sind sie jedoch eindeutig genug, um korrekt umgesetzt zu werden.

2. Die Zielsprache ist eine proprietäre Skriptsprache. Sie hat klar definierte Syntax und Semantik (ANMERKUNG NIELS: HIER BIETEN SICH AUCH BEGRIFFE WIE REGULÄRE SPRACHEÖDER "KONTEXTFREIE SPRACHE"(JE NACHDEM, WAS ES IST) NACH NOAM CHOMSKY AN.) und kann prinzipiell automatisch erzeugt werden.

3. Die Erfassung der Validierungslogik muss zwingend durch die Auftraggeber erfolgen, da sie die fachliche Verantwortung für die digitalen Antragsformulare tragen. Eine direkte Umsetzung durch Entwickler ohne vorliegende Anforderung ist weder organisatorisch noch inhaltlich möglich.

4. Die Qualität der automatischen Übersetzung muss so hoch sein, dass die zusätzliche Prüf- und Testlast akzeptabel bleibt. (ANMERKUNG NIELS: ICH DENKE, DAS IST NICHT GEGENSTAND DEINER ARBEIT, ODER? DANN IST ES AUCH NICHT TEIL DER PROBLEMSTELLUNG.)

Vor diesem Hintergrund untersucht die vorliegende Arbeit, ob und unter welchen Bedingungen KI-gestützte Verfahren geeignet sind, die beschriebene Übersetzungsaufgabe zuverlässig zu automatisieren. Die konkreten Forschungsfragen, Hypothesen und Zielgrößen werden im nächsten Kapitel definiert.

(TODO: DAS FOLGENDE GGF. NOCH IN DIESEM KAPITEL EINSORTIEREN:)

2.4 Kontext

```
Auf Basis des Prototypen soll in Zukunft ein Werkzeug entwickelt
werden, das Entwicklern das Implementieren von Kundenvorgaben aus dem
Formularverwalter abnimmt. Es gibt bereits für gewisse Vorgaben-Arten
Werkzeuge, die das tun.

Für die Übernahme von Plausibilitätsprüfungen gibt es noch kein
Werkzeug. Das Werkzeug soll als Eingabe den Vorgaben-Namen bekommen.
Als Ausgabe soll es das Skript anpassen, das die Vorgabe umsetzt. Dafür
soll es:
1. Auf Basis des Vorgabennamen die Veränderung der Vorgabe per API aus
dem Formularverwalter abfragen
2. Im Webclient-Code ein oder mehrere Skripte auffinden, welche die
Vorgabe umsetzen
3. Ggf. weitere Kontextinformationen aus dem Formularverwalter oder
Webclient Code laden
4. Auf Basis der veränderten Vorgabe, des/r ursprünglichen Skripte(s)
und ggf Kontextinformationen ein neues Skript generieren
5. Das veränderte Skript durch kompilieren auf syntaktische Korrektheit
prüfen und ggf zurück zu Schritt 4
6. Das veränderte Skript im Webclient-Code ablegen
```

3 Ziele

(GEPLANTE SEITENZAHL: 3 - 6 | T: 23.11.25)

```
Leitfragen

Was sind die operativen Ziele der Arbeit (z. B. Prototyp,
Evaluationskriterien, Metriken)?

Welche forschungsleitende Hypothese(n) werden getestet?

Welche messbaren Erfolgskriterien definierst du (z. B. Genauigkeit,
Zeitersparnis, Verständlichkeit, Fehlerreduktion)?

Welche Nebenbedingungen gelten (z. B. verwendete DSL, Datengrundlage,
Datenschutz)?

Welche Annahmen werden explizit getroffen und wie werden sie
überprüft?
```

Im Zentrum dieser Arbeit steht das Generieren eines veränderten Skriptes. Auf Basis einer angepassten Kundenvorgabe. Dafür wird explorativ untersucht, mit welchen KI-Ansätzen die Änderung der Kundenvorgaben am präzisesten in Skripte übersetzt werden können.

Dabei werden verschiedene Modelle, Prompting-Methoden und Kontextinformationen getestet, um die Auswirkungen auf Kompilierbarkeit, Testfall-Erfolg und Codequalität zu bewerten.

3.1 Forschungsziele

Welche Konfiguration aus Prompting-Methode, KI-Modell und Kontextinformationen („Prototyp“) erzielt bei der Anpassung von Skripten auf Basis von Kundenvorgaben die höchste syntaktische und semantische Korrektheit?

Zur Beantwortung der Forschungsfrage sollen folgende Hypothesen überprüft werden:

1. Teurere OpenAI Modelle, liefern bessere syntaktische und semantische Korrektheit sowie geringere Komplexität als günstigere.
2. Mehr Beispiele im Kontext erhöhen die syntaktische Korrektheit.
3. Relevante Formulardefinitionen im Kontext erhöhen die semantische Korrektheit.
4. API und JDOC im Kontext erhöhen die syntaktische und semantische Korrektheit und verringern die Komplexität.
5. Chain-of-Thought-Prompting verbessert die Korrektheit von non-reasoning Modellen.

3.2 Operative Ziele

In der Arbeit sollen drei Artefakte entstehen: Testdatensatz-Sammlung, Evaluations-Pipeline und Prototypen.

3.2.1 Testdatensatz-Sammlung

Es soll eine Testdatensatz-Sammlung entstehen, die mindestens 30 **(TODO: QUELLE RAUSSUCHEN, WARUM 30 DIE MAGISCHE UNTERGRENZE IST)** hat. Ein Testdatensatz in dieser Sammlung soll enthalten:

- Die Kundenvorgabe vor der Anpassung
- Die Kundenvorgabe nach der Anpassung
- Das Skript vor der Anpassung
- Das Skript nach der Anpassung (Referenzwahrheit)
- Teststories, die das Skript nach der Anpassung validieren

3.2.2 Evaluations-Pipeline

Diese Pipeline soll für jeden Eintrag in der Testdatensatz-Sammlung mit einer Prompting-Methode von einem KI-Modell das „Skript nach der Anpassung generieren“ lassen. Die generierten Skripte sollen evaluiert und ein Bericht über die Ergebnisse erstellt werden.

Der Bericht soll folgende Metriken enthalten:

Syntaktische Korrektheit Ein Skript ist syntaktisch korrekt, wenn es kompilierbar ist.

Semantische Korrektheit Ein Skript ist semantisch korrekt, wenn es alle Testfälle erfüllt.

Komplexität Komplexität des Skriptes im Verglich zur Referenzwahrheit.

Kosten Preis (in Euro) für die Generierung eines Skriptes.

3.2.3 Prototypen

Ein Tripel aus Prompting-Methode, KI-Modell und Kontext-Information stellt einen Prototypen dar.

Im folgenden sind die Varianten der einzelnen Bestandteile aufgelistet, die zu Prototypen kombiniert werden sollen:

Prompting-Methoden

- Zero-Shot-Prompting
- One-Shot-Prompting
- Few-Shot-Prompting
- Chain-of-Thought-Prompting

KI-Modelle

- GPT-5-nano (kleines Reasoning-Modell)
- GPT-5.1 (großes Reasoning-Modell)
- GPT-5.1-Codex (Modell spezialisiert auf agentic coding)
- GPT-4.1 („Klügstes nicht-reasoning-Modell“)

Kontext-Informationen

- Keine weiteren Kontextinformationen
- Formulardefinition der relevanten Formulare
- API und JDOC der zur Verfügung stehenden Funktionen

3.3 Annahmen und Nebenbedingungen

Die deg stellt zur Durchführung dieser Arbeit Zugang und Kapital für die Verwendung von OpenAI Modellen bereit. Deshalb werden diese verwendet.

Die Skripte werden in der domänenspezifischen Sprache „formScript“ umgesetzt. Diese ist außerhalb der deg unbekannt und dadurch nicht in den Trainingsdaten der KI-Modelle vorhanden. Sie wird bis 2028 verwendet werden. Es ist geplant, dass die Skripte im Jahr 2028 im Rahmen einer technologischen Modernisierung in die Skriptsprache TypeScript migriert werden. Es wird davon ausgegangen, dass das Generieren syntaktisch korrekter Skripte dadurch einfacher wird und die semantische Korrektheit eine Herausforderung bleibt.

Als Basis zur Erstellung der Testdatensatz-Sammlung wird der aktuelle Daten zugegriffen. Vorgaben und ihre Änderungen werden per API aus dem produktiven System des Formularverwalters abgerufen. Skripte und ihre Änderungen werden aus der Versionskontrolle des Sourcecode-Repositories des Webclients entnommen. Es wird davon ausgegangen, dass sich aus diesen Datenbeständen

ausreichend Testdatensätze extrahieren lassen. Dafür wird von drei Voraussetzungen ausgegangen:

1. Die Zuordnung von Formularverwalter-Vorgaben zu Webclient-Skripten ist in ausreichender Qualität und Quantität möglich.
2. Es liegen ausreichend Skripte im Webclient vor, für die Teststories definiert sind.
3. Die Teststories sind von ausreichender Qualität, sodass das Erfüllen der Teststories ein semantisch korrektes Skript bedeutet.

4 Forschungsstand

(GEPLANTE SEITENZAHL: 6 - 9 | T: 07.12.25)

Leitfragen

Welche relevanten Disziplinen sind einzubeziehen (NLP, Requirements Engineering, DSL-Design, HCI, Software-Engineering)?

Welche bestehenden Ansätze zur Übersetzung natürlicher Sprache in Code/DSL gibt es?

Wie ist der Stand der Technik für Plausibilitätsprüfungen in Formularen?

Welche KI-Modelle/Methoden wurden bisher eingesetzt und mit welchem Erfolg?

Welche Lücken oder offene Fragen identifiziert die Literatur, die deine Arbeit füllen soll?

Welche Bewertungsmetriken und Benchmarks werden in der Literatur genutzt?

Wo widersprechen sich Befunde? Welche Positionen sind kontrovers?

5 Methode

(GEPLANTE SEITENZAHL: 9 - 12 | T: 04.01.26)

```
Leitfragen

Welchen methodischen Ansatz wählst du (gestaltungsorientiert: Design
Science, Prototyping, Mixed-Methods, Usability-Testing)?

Welche Artefakte wirst du entwerfen (DSL-Erweiterung,
Konversionspipeline, Nutzerinterface, Evaluationsskripte)?

Welche Datengrundlage verwendest du (Anwendungsfälle,
Formularbeispiele, Annotatoren, synthetische Daten)?

Wie sieht der Entwicklungsworkflow aus (Iterationen,
Validationsschritte, Tools, Versionierung)?

Welche KI-Modelle und Prompt-/Fine-tuning-Strategien werden eingesetzt?

Welche Metriken und Messmethoden nutzt du zur Bewertung
(Precision/Recall, BLEU/ROUGE ungeeignet? -- nenne passendere Metriken:
Genauigkeit der Regelübersetzung, menschliche Lesbarkeit, Zeitaufwand)?

Welche Nutzerstudien oder Experten-Reviews planst du (Anzahl,
Auswahlkriterien, Aufgaben, Messinstrumente)?

Wie stellst du Reproduzierbarkeit und Validität sicher (Artefakte,
Skripte, Protokolle, Datenschutz)?

Welche ethischen und rechtlichen Fragen beachtest du (Bias,
Datenherkunft, Verarbeitung personenbezogener Daten)?
```

5.1 Plausibilitätsprüfungen

Für eine Live-Validierung (auch Plausibilitätsprüfung genannt) werden im Formularverwalter folgende Eigenschaften definiert: Name, Fehlertext, Beschreibung, Fehlerlevel und verknüpfte Formulare und Antragsverfahren.

Name

Der Name ist der Bezeichner der Vorgabe. Er ist nicht eindeutig – es kann mehrere Vorgaben mit dem gleichen Namen geben. Die Auftraggeber sind dazu aufgefordert die Namen technisch auswertbar zu gestalten, indem sie auf die Verwendung von Umlauten, Sonderzeichen und Leerzeichen im Namen verzichten. Das wird nicht erzwungen und deshalb nicht bei allen Vorgaben berücksichtigt.

Beispiel: *IBAN_Laenge*

Fehlertext

In diesem Feld wird der Text eingetragen, der dem Anwender angezeigt werden soll. Meistens wird genau der Text dem Anwender angezeigt. Manchmal werden Platzhalter verwendet. Wie diese zu ersetzen sind, geht meistens aus der Beschreibung hervor.

Beispiel: *Die von Ihnen eingegebene IBAN hat nicht die erforderlichen 22 Stellen.*

Beschreibung

Die Beschreibung enthält die Logik, die bestimmt, unter welchen Bedingungen die Meldung ausgegeben werden soll. Die Logik wird in natürlicher Sprache beschrieben. Die Qualität der Beschreibungen variiert stark – abhängig von der Komplexität der Prüfung und dem Hintergrund des Autors.

In der Beschreibung können Felder aus den Verknüpften Formularen referenziert werden. Wenn das gemacht wird, kann die Referenz technisch zu einem Feld im Formular zugeordnet werden. In manchen Fällen wird nur der technische Bezeichner eines Formularfeldes (wie *bi_iban* im unten stehenden Beispiel) oder nur eine natürlichsprachliche Umschreibung (bspw. „In der ersten Spalte der Tabelle“) verwendet.

Beispiel *bi_iban: Wenn es sich um eine DE - IBAN handelt, muss sie incl. DE genau 22 Zeichen lang sein.*

Fehlerlevel

Das Fehlerlevel ist der Schweregrad der Prüfung. Es reicht von Hinweis bis Disqualifikation. Es wird über ein Auswahlfeld im Formularverwalter erfasst. Es ist dadurch eindeutig und technisch auswertbar. Es gibt 5 Level:

Information, Warnung, Fehler Diese Level sind rein informativ. Die Meldungen werden angezeigt, schränken die Bearbeitung aber nicht weiter ein.

Fataler Fehler Enthält der Antrag ein Formular mit einem fatalen Fehler, kann er nicht eingereicht werden.

Disqualifikation Ein Disqualifikationsfehler schließt ein Formular vom Antrag aus. Der Antrag kann eingereicht werden, aber das entsprechende Formular wird nicht mit eingereicht.

Beispiel: *Fataler Fehler*

verknüpfte Formulare und Antragsverfahren

Eine Plausi-Vorgabe ist mit den Antragsverfahren verknüpft, für die sie gelten soll. Und sie ist mit den Formularen Verknüpft, auf die für die Prüfung zugegriffen werden muss.

Beispiel:
Antragsverfahren: Brandenburg Agrar-Antrag 2026, Brandenburg ELER-Antrag 2026, Mecklemburg-Vorpommern Agrar-Antrag 2026, Mecklemburg-Vorpommern ELER-Antrag 2026, Schleswig Holstein Agrar-Antrag 2026
Dokument: Stammdaten

5.2 Implementierung

Diese Vorgaben werden von der data experts mit Hilfe der proprietären Skriptsprache „formScript“ implementiert. Diese ist für die Verarbeitung und Manipulation von Formulardaten geschaffen worden. Die Skripte werden als Exten-

ded Markup Language (XML)-Dateien im Quellcode hinterlegt. Zur Compile-Zeit werden sie zu Java-Code übersetzt. Quellcode 5.1 ist das Skript zu der Vorgabe „IBAN_Laenge“, die oben als Beispiel verwendet wurde.

Quellcode 5.1: IBAN_Laenge Skript

```
<?xml version="1.0" encoding="UTF-8"?><scripts>
<script name="MC26_STDA_1_IBAN_Laenge"><![CDATA[
@meta name: "MC26_STDA_1_IBAN_Laenge";

master stda1 as #202617101;

var iban = stda1->bi_IBAN@value;
if(Strings.startsWith(iban, "DE")
&& Strings.length(iban) != 22) {
        fatal [stda1->bi_IBAN]
        : "Die von Ihnen eingegebene IBAN hat nicht die
        erforderlichen 22 Stellen."
        : meta {errorId: "IBAN_Laenge"};
}]]></script>
</scripts>
```

6 Ergebnisse

(GEPLANTE SEITENZAHL: 9 - 15 | T: 18.01.26)

Leitfragen

Welche Artefakte sind entstanden (DSL-Spezifikation, Konverter, Beispielskripte, Evaluationsdatensatz)?

Welche quantitativen Ergebnisse liegen vor (Metriken, Tabellen, statistische Tests)?

Welche qualitativen Ergebnisse liegen vor (Expert:innen-Feedback, Fallstudien, Fehlerklassen)?

Wie verhalten sich die Ergebnisse gegenüber den definierten Erfolgskriterien?

Welche typische Fehlerfälle/Fehlermodi traten auf? Gib konkrete Beispiele.

Welche Designentscheidungen haben sich als besonders wirkungsvoll erwiesen?

Welche Artefakte oder Prototyp-Versionen sind reproduzierbar und wo liegen Einschränkungen?

7 Diskussion

(GEPLANTE SEITENZAHL: 9 - 12 | T: 25.01.26)

Leitfragen

Was bedeuten die Ergebnisse für die Hypothese? Bestätigt, modifiziert oder widerlegt sie die Hypothese?

Welche Ursachen haben beobachtete Fehler oder Abweichungen? Technisch und methodisch getrennt.

Wie verallgemeinerbar sind die Ergebnisse (Kontexte, Domains, Formulartypen)?

Welche Implikationen haben die Ergebnisse für Praxis (Implementierungsempfehlungen) und Forschung (neue Fragen)?

Welche Limitationen hatte die Studie (Datenmenge, Expert:innenzahl, Modellgrenzen, Metrik-Validität)?

Wie würden alternative Methoden oder andere DSL-Designs das Ergebnis wahrscheinlich verändern?

Welche Risiken und Nebenwirkungen (z. B. Fehlinterpretation durch Fachanwender, Maintenance-Aufwand) sind relevant?

8 Fazit

(GEPLANTE SEITENZAHL: 3 - 6 | T: 01.02.26)

Leitfragen

Was sind die knappsten, prüfungsrelevanten Erkenntnisse der Arbeit?

Welche konkreten Empfehlungen gibst du für Entwickler, Behörden oder Forscher?

Welche offene Forschungsfragen bleiben und welche nächsten Schritte empfiehlst du (konkrete Experimente, Skalierung, Integration)?

Welche praktische Bedeutung hat die Arbeit kurz- und mittelfristig?

Welche Learnings für dein eigenes Vorgehen sind wichtig (Was würdest du beim nächsten Mal anders machen)?

Literaturverzeichnis

data experts gmbh: Impressum » data experts gmbh. https://www.data-experts.de/impressum/. Letzter Zugriff: 2024-02-23.

Brandenburg, den 31. August 2025

Hiermit versichere ich, FLORIAN LEMNITZER, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt habe und dass die Arbeit in gleicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

FLORIAN LEMNITZER