SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Controlling the Evolution of a Legal Knowledge Base for an In-Context Learning AI System - Design and Implementation

Colin John Hubertus Graf von Hardenberg

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Controlling the Evolution of a Legal Knowledge Base for an In-Context Learning AI System - Design and Implementation

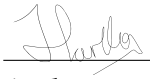
Steuerung der Entwicklung einer juristischen Wissensbasis für ein kontextabhängig lernendes KI-System - Entwurf und Implementierung

Author:	Colin John Hubertus Graf von Hardenberg
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	M.Sc. Oliver Wardas
Submission Date:	12.12.2024

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 11.12.2024

Location, Submission Date



Author

AI Assistant Usage Disclosure

Introduction

Performing work or conducting research at the Chair of Software Engineering for Business Information Systems (sebis) at TUM often entails dynamic and multi-faceted tasks. At sebis, we promote the responsible use of *AI Assistants* in the effective and efficient completion of such work. However, in the spirit of ethical and transparent research, we require all student researchers working with sebis to disclose their usage of such assistants.

For examples of correct and incorrect AI Assistant usage, please refer to the original, unabridged version of this form, located at this link.

Use of *AI Assistants* for Research Purposes

I have used AI Assistant(s) for the purposes of my research as part of this thesis.

X Yes ☐ No ☐

Explanation: I have used Grammarly and ChatGPT for spelling corrections and grammar improvements. For translation purposes, I have used DeepL. For debugging, I have used ChatGPT and Copilot.

I confirm in signing below, that I have reported all usage of AI Assistants for my research, and that the report is truthful and complete.

Munich, 11.12.2024	
Location, Date	Author

Acknowledgments

A sincere thank to my supervisor, Professor Dr. Matthes, and the entire Chair of Software Engineering for Business Information Systems that I was able to write my thesis under their guidance. I am especially grateful to my advisor, Oliver Wardas, for recognizing my interest in Legal Tech and offering valuable support throughout the course of this thesis.

Abstract

This thesis focuses on further developing the KIBeKodA platform, designed to improve law professional's workflow when analyzing employment contracts. The primary objectives were to implement a versioning system for examination guidelines to enhance the transparency in clause classification and establish intuitive analytics so that lawyers can improve the systems without technical knowledge.

Starting with a comprehensive literature review, we provide an overview on how the KIBeKodA system works and how different types of Slowly Changing Dimensions (SCD) are implemented according to best practices. Further, we investigate different metrics that are possibly applicable and help the law professionals to maintain the system themselves. The literature review serves as a basis for the implementations and as a guideline for the successful implementation of the desired functionality. A questionnaire was also developed to assess the system's usability and intuitiveness in future studies.

The results demonstrate that the enhanced KIBeKodA system provides a user-friendly interface and enables legal professionals to optimize their examination guidelines, improving classification outcomes. The prepared questionnaire offers a practical tool for unmoderated usability studies to further refine the system.

However, certain limitations were identified, including overly lenient classifications and the need to refine specific metrics further. Despite these difficulties, the system successfully addressed the initial lack of transparency and provided a solid foundation for future enhancements. The knowledge acquired paves the way for further development of the KIBeKodA system, guaranteeing that it will continue to be a useful tool for attorneys.

Kurzfassung

Diese Arbeit konzentriert sich auf die Weiterentwicklung der KIBeKodA-Plattform, die entwickelt wurde, um die Arbeitsabläufe von Juristen bei der Analyse von Arbeitsverträgen zu verbessern. Die Hauptziele waren die Implementierung eines Versionierungssystems für Prüfanleitungen, um die Transparenz bei der Klauselklassifikation zu erhöhen, sowie die Entwicklung intuitiver Analysefunktionen, damit Anwälte das System ohne technische Vorkenntnisse optimieren können.

Ausgehend von einer umfassenden Literaturrecherche wird ein Überblick über die Funktionsweise des KIBeKodA-Systems gegeben, einschließlich der Implementierung verschiedener Typen von Slowly Changing Dimensions (SCD) nach Best Practices. Darüber hinaus wurden potenziell anwendbare Metriken untersucht, die Juristen bei der eigenständigen Wartung und Optimierung des Systems unterstützen. Die Literaturrecherche diente als Grundlage für die Implementierungen sowie als Leitfaden für die erfolgreiche Umsetzung der gewünschten Funktionalitäten. Zur Bewertung der Implementierungen wurde zudem ein Fragebogen entwickelt, der das System hinsichtlich seiner Intuitivität und Benutzerfreundlichkeit evaluiert.

Die Ergebnisse zeigen, dass das verbesserte KIBeKodA-System eine benutzerfreundliche Oberfläche bietet und es Juristen ermöglicht, ihre Prüfanleitungen zu optimieren, was zu besseren Klassifikationsergebnissen führt. Der vorbereitete Fragebogen stellt ein praktisches Werkzeug für unmoderierte Evaluierungen dar, um das System weiter zu verfeinern.

Es wurden jedoch auch bestimmte Einschränkungen identifiziert, darunter zu großzügige Klassifikationen und die Notwendigkeit, bestimmte Metriken weiter zu verfeinern. Trotz dieser Herausforderungen konnte das System die anfänglichen Transparenzprobleme erfolgreich beheben und eine solide Grundlage für zukünftige Verbesserungen schaffen. Die gewonnenen Erkenntnisse bilden die Basis für eine kontinuierliche Weiterentwicklung des KIBeKodA-Systems, um sicherzustellen, dass es eine wertvolle Unterstützung für Juristen bleibt.

Contents

1. Introduction

The Legal Tech industry has undergone significant transformations in recent years, with Artificial Intelligence (AI) applications becoming increasingly common and demanded [1]. This thesis is located within the broader context of the research project AI-Assisted Legal Analysis and Correction of German Employment Contracts (KIBeKodA), which aims to leverage AI capabilities in the legal domain.

1.1. Motivation of the Project

The Legal Tech market is experiencing remarkable growth, with the global market size reaching $26.7 billion in 2024. Projections indicate further expansion, with estimates suggesting a compound annual growth rate (CAGR) of 10.2% by the end of this decade [2]. More optimistic forecasts predict the market will more than double, growing from $31.59 billion to $63.59 billion by 2032 [3]. Another indicator is the amount of publications in the legal tech domain on Google Scholar growing from 391 articles published in 2017 to 1550 in 2023. In December 2024 the number of publications is at 1360 for the year.

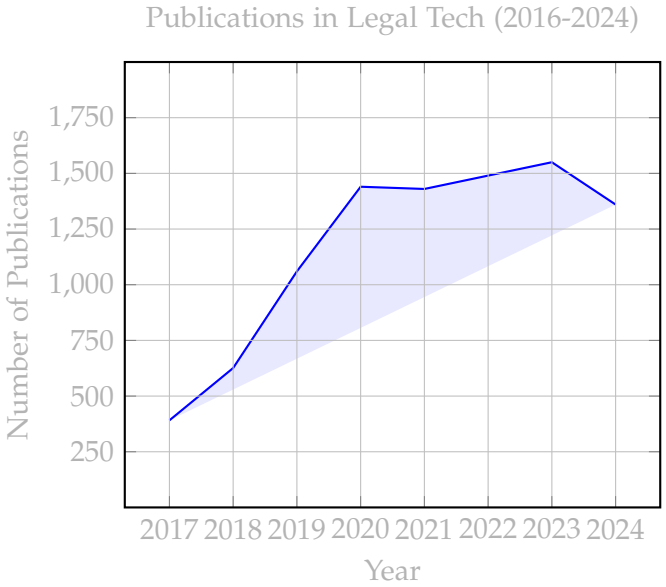


Figure 1.1.: Number of publications from 2017 to 2024 for the query: "legal tech" (as of 10.12.2024)

This growth underscores technological solutions' increasing importance and adoption in legal practices.

An up-and-coming area within Legal Tech is the development of Contract Lifecycle Management (CLM) tools. These tools enable rapid and legally compliant contract creation and analysis, often using AI. Gartner identifies conversational AI and chatbots as "immature but expected to be deployed widely" in legal technology [4]. The integration of a Large Language Model (LLM) into legal processes offers the potential for significant reductions in time and personnel costs for legal firms [4].

1.1.1. The KIBeKodA Project

The KIBeKodA software, developed as part of a research project at the Technical University of Munich (TUM), exemplifies the application of AI in legal contract analysis. This system focuses on reviewing and analyzing employment contracts using a combination of LLMs and human expertise. The AI assessments are verified by lawyers, ensuring accuracy and reliability. A crucial component of the KIBeKodA system is using lawyer-written review rules, so-called examination guidelines (Prüfanleitungen). These guidelines are essential in helping the LLM evaluate individual clauses more effectively. However, these rules are subject to change due to various factors such as semantic deficiencies, poor text quality that hinders AI utilization, or changes in legislation. The components of the current system can be seen in Figure 1.2.

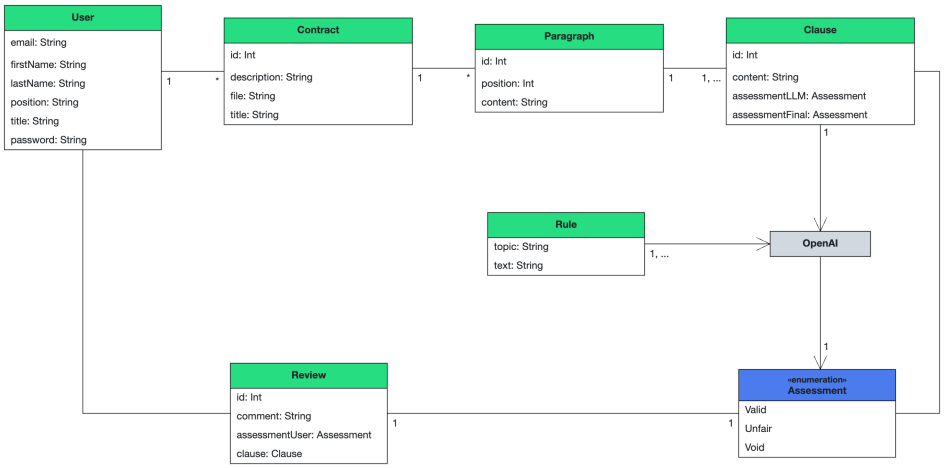


Figure 1.2.: Current components of KIBeKodA

1.1.2. Integration into the Project

The main focus of this thesis is to address a gap in the current KIBeKodA system: the lack of traceability in the evolution and application of examination guidelines. Currently, there is no mechanism to track or trace changes made to the rules, nor is it possible to determine

which specific examination guidelines are being utilized by the LLM in its analysis of contract clauses. If rule changes are not tracked, it becomes necessary to recheck all previously analyzed contracts to determine where a rule might have been relevant. The thesis aims to address these problems by designing and implementing new software components as well as modifying existing ones within the KIBeKodA system. The focus will be on developing an intuitive user interface that allows non-technical users to manage the system, track examination guideline changes, and visualize key performance metrics for AI evaluations. The lawyers shall be able to optimize the platform and rules without technical help, but even if there are analytical metrics available, they are not helpful unless they are presented in a way that is easily understandable and useable for non-technicians. By doing so, this work objective is to enhance the transparency, manageability, and overall effectiveness of the KIBeKodA system.

1.2. Outline

The thesis begins by providing relevant background information to the reader required for the following chapters, which will contain details of the methodology, implementation, and results of the solutions, as well as discuss its implications for optimizations in the future.

2. Background

To provide a comprehensive overview, we will explain the fundamentals of the current KIBeKodA system and its components. Furthermore, we will present technical basics to the reader. This background information is crucial for understanding the context of the problem and the proposed solutions.

2.1. Current System

The KIBeKodA system is built on the robust and widely adopted MERN technology stack. This stack is a popular choice in web development due to its flexibility, performance, and extensive community support. According to the annual developer survey by Stack Overflow, all components of the MERN stack consistently rank among the top five most-used web development technologies [5]. The MERN stack consists of the following components [6]:

- **M**ongoDB: A NoSQL database system allowing flexible data storage without a fixed schema.
- **E**xpress: A web application framework for Node.js that simplifies the process of building web applications.
- **R**eact: A JavaScript library for building user interfaces, known for its efficiency and component-based architecture.
- **N**ode.js: A JavaScript runtime, enabling server-side execution of JavaScript.

In general, the KIBeKodA system can be divided into two parts:

- Backend: Responsible for data processing, storage, and communication with the frontend.
- Frontend: Executed on the user's device, providing an interactive and intuitive user interface.

2.1.1. Backend

In the backend, we distinguish between the application server, the central part of the system, and the classification service, which handles the LLM classification and generation of assessments. Additionally, we have the formulation service, which deals with finding template clauses to replace invalid or unfair clauses, and finally, we have the database, which is responsible for storing the information.

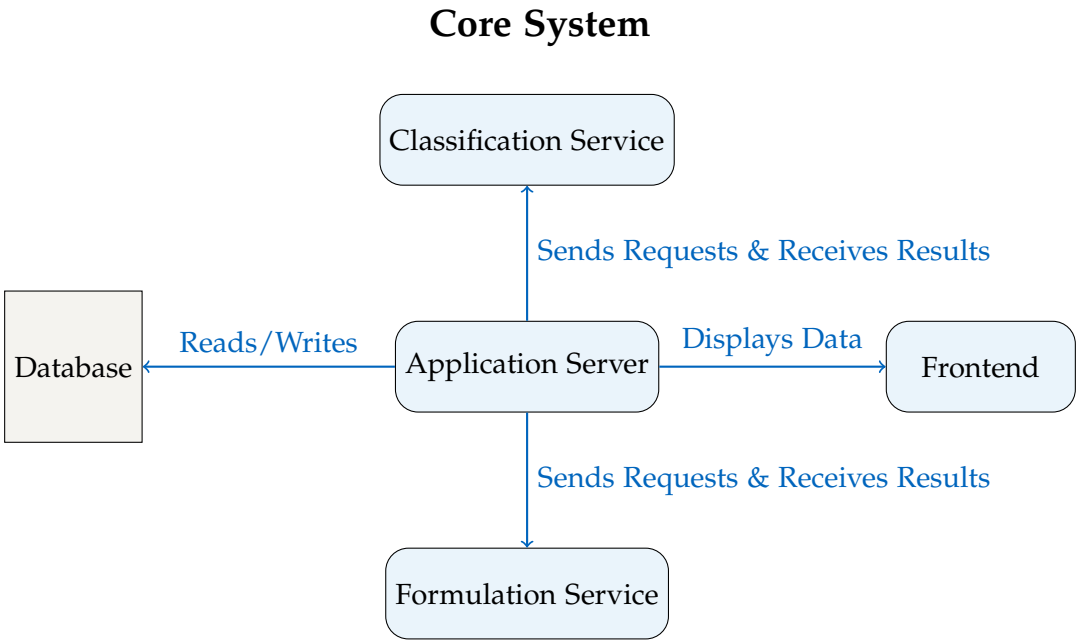


Figure 2.1.: System architecture of the main components

The heart of the backend is the application server, which manages all interactions between different parts of the system. The application server is structured around three essential elements: controllers, services, and models. Controllers are responsible for handling incoming requests from the frontend. They act as intermediaries, routing these requests to the appropriate services based on the type of action required (e.g., retrieving user data or processing a contract). Services contain the core logic of the application. They perform operations such as creating, updating, or deleting data and interacting with database models to ensure data is processed correctly. Models, in turn, define how data is structured within the database. They represent key entities in the system, such as users or contracts and ensure that data is stored and retrieved consistently.

An important part is the classification service, which evaluates clauses and creates LLM assessments using a tailored prompt with all current examination guidelines and a contract clause. Another critical component is the formulation service, which addresses situations where contract clauses are classified as unfair or invalid. This service then finds relevant template clauses that can replace the problematic ones to ensure the compliance of a contract.

The backend also includes the database that stores all the persistent data for the system. The database, i.e., MongoDB, works closely with the models to ensure consistent data that can be stored and retrieved.

2.1.2. Frontend

The frontend is responsible for the user interface and interactions with the system's user. It communicates with the backend with the help of API calls, which send requests to the

backend and receive data to display or process actions initiated by the user. Each page, i.e., users or template clauses, has a set of User Interface (UI) components that are used to display information for the concrete page. Another important aspect is the features that handle the specific page's API calls. For example, the contract feature manages all interactions related to contracts — loading contract data from the backend, displaying it to the user, and handling any updates or modifications made by the user.

2.2. Technological Basics

To further improve understanding, we will explore technologies at a superficial level that are used and will be relevant to this work.

2.2.1. Artifical Intelligence (AI)

AI is a technology that has a significant influence on modern society and will be even more important in the future [7]. Defined by the Oxford English Dictionary as "the capacity of computers or other machines to exhibit or simulate intelligent behaviour" [8], AI has evolved from a theoretical idea in the mid-20th century to a reality that is reshaping industries and social structures. Alan Turing, a pioneer in the entirety of computer science, envisioned in the late 1940s that the ultimate goal of AI would be a machine capable of learning from its own experience. This idea continues to guide AI research today [9]. AI involves the development of systems capable of completing tasks that typically require human intelligence, such as visual perception, speech recognition, and decision-making. However, not all AI systems are designed to achieve this breadth of capabilities. Therefore, we can broadly divide AI into two classes.

The first is weak AI, also known as narrow AI. It performs tasks like a human in a predefined environment. While efficient and maybe even better than humans in their specific domains, these systems lack general intelligence and cannot apply their capabilities beyond their specified functions. The other is strong AI, which would be the equivalent of programming an AI to act like a human mind, think, be intelligent, and have beliefs and wishes normally described as human. Although research in AI is advancing, it remains a concept rather than a practical reality [10]. If "strong AI systems may be able to continue to program themselves smarter at ever-increasing, exponential speeds" [11], it might evolve into Artificial Superintelligence (ASI) - a third, yet not thinkable, category, which could have an unimaginable impact on human life [11].

2.2.2. Large Language Model (LLM)

Large Language Models are advanced AI systems capable of understanding and generating human-like text [12]. These models represent a significant breakthrough in Natural Language Processing (NLP) and have revolutionized various applications, from chatbots and virtual assistants to content generation and research assistance [12]. LLMs are trained on vast datasets with billions of entries, making them highly versatile. In the KIBeKodA project, a

large language model is used in several steps, such as evaluating clauses as well as finding template clauses.

2.2.3. Natural Language Processing (NLP)

Natural Language Processing, or NLP, is a part of machine learning that enables "computers and digital devices to recognize, understand, and generate text and speech by combining computational linguistics" [13]. NLP can be divided into two main categories: Natural Language Understanding (NLU) and Natural Langauge Generation (NLG) [14]. NLU focuses on interpreting human language input, whether in text or speech form, by identifying elements such as emotions, concepts, and other semantic features [14]. NLG, on the other hand, takes the processed information from NLU and generates coherent and contextually appropriate language output. This process involves identifying goals, planning how to achieve these based on the situation and available communicative sources, and realizing the plans as text [14]. NLP is widely used today in various applications, for example:

- Text Classification: Labeling emails as spam or performing document analysis
- Autocomplete Suggestions: Predicting search queries based on user input
- Conversational Systems: Powering chatbots and virtual assistants

In the context of the KIBeKodA system, NLP is applied to understand and classify contract clauses.

2.2.4. In-Context Learning (ICL)

In-Context Learning (ICL) refers to an AI's ability to learn from examples provided within a given prompt without requiring further training. It leverages pre-trained models like LLMs to adapt to new tasks based on contextual information. This allows the model to generalize from information provided at runtime, enabling it to perform tasks such as classification, translation, or summarization without explicit retraining [15].

For instance, an LLM can be given a few guidelines on how to classify contract clauses as valid or invalid within the prompt itself, and it will apply this logic to clauses it encounters during inference.

2.2.5. Optical Character Recognition (OCR)

OCR is a technology that converts different types of documents - such as PDFs, images, or scanned paper materials - into machine-readable text [16]. OCR plays a critical role in KIBeKodA by converting scanned contracts or PDFs into digital text for further processing and analysis by the system. The process involves four steps [17]:

- Image Acquisition: Document pages are gathered and converted into high-contrast black-and-white images.

- Pre-processing: Enhancing image quality for better recognition.
- Text & Character Recognition: Dark areas in the image are processed, mapping letters, digits, and symbols.
- Post-processing: Correcting errors and structuring the recognized text into usable formats.

At this stage, the LLM employs Natural Language Processing to use the OCRs output and structures data into clauses and paragraphs, providing output that lawyers can use more effectively.

2.3. Workflow

For a better understanding of a typical workflow in the KIBeKodA system, we will give an example and lead the reader through it.

Lawyer Mr. Doe uploads a contract for the company Smith. This contract is then digitized using OCR technology, where the system then receives the contract of firm Smith in a JSON format. This data structure, in combination with Mr. Does name and further metadata is stored in MongoDB. Automatically, the clauses of the contract are sent to the classification service, where each clause, in combination with all existing examination guidelines and a tailored prompt, is used as input and sent to the LLM. The Model then classifies the clause and returns an output where the assessment "valid", "unfair" or "invalid" is assigned to it. This will be repeated for every clause in the contract. These classifications, in combination with the contract and its paragraphs and clauses, are then displayed in the UI for the lawyer. Mr. Doe checks each LLM assessment and classifies the clause himself with the named possibilities. Once the lawyer supervises all assessments, "unfair" and "invalid" clauses can be replaced in the next step by finding suitable template clauses that are written by the lawyers of Mr. Does law firm and can be adjusted according to the special needs of the current situation. The contract with the updated clauses can be downloaded in the final step.

This workflow, also seen in Figure 2.2, demonstrates integrating advanced technologies like OCR and LLMs with legal expertise to create a robust contract analysis and modification tool. In the current workflow, examination guidelines are used to guide the LLM in evaluating and classifying clauses within contracts. However, one significant limitation is the lack of transparency about which specific examination guideline is applied to each clause during evaluation. This lack creates a challenge when legal texts change. For example, if a contract is initially evaluated with the instruction "Der Brutto Mindestlohn beträgt 9.82€." ("The gross minimum wage is €9.82."), and later the legal text updates, thus leading to a change in the rule to "Der Brutto Mindestlohn beträgt 10.45€." ("The gross minimum wage is €10.45."), all analyzed contracts would need re-evaluation, since we do not know in which contract and clause this rule was applied. Such a process directly contradicts the project's aim of reducing time and personnel costs.

Another significant issue is the current shortage of insight into the LLM performance. We do not have data on how accurately it applies individual examination guidelines, nor on the

frequency with which its assessments require correction by legal professionals. Knowing the LLM's accuracy in applying these guidelines would help identify which examination guidelines may need refinement to improve overall system performance. This bachelor thesis aims to address these issues by developing a system that tracks and adapts to changes in examination guidelines and provides transparent, understandable information on the LLM's rule applications. Additionally, a core objective is to design a user-friendly interface that allows lawyers to monitor and refine the system independently, regardless of their technical proficiency.

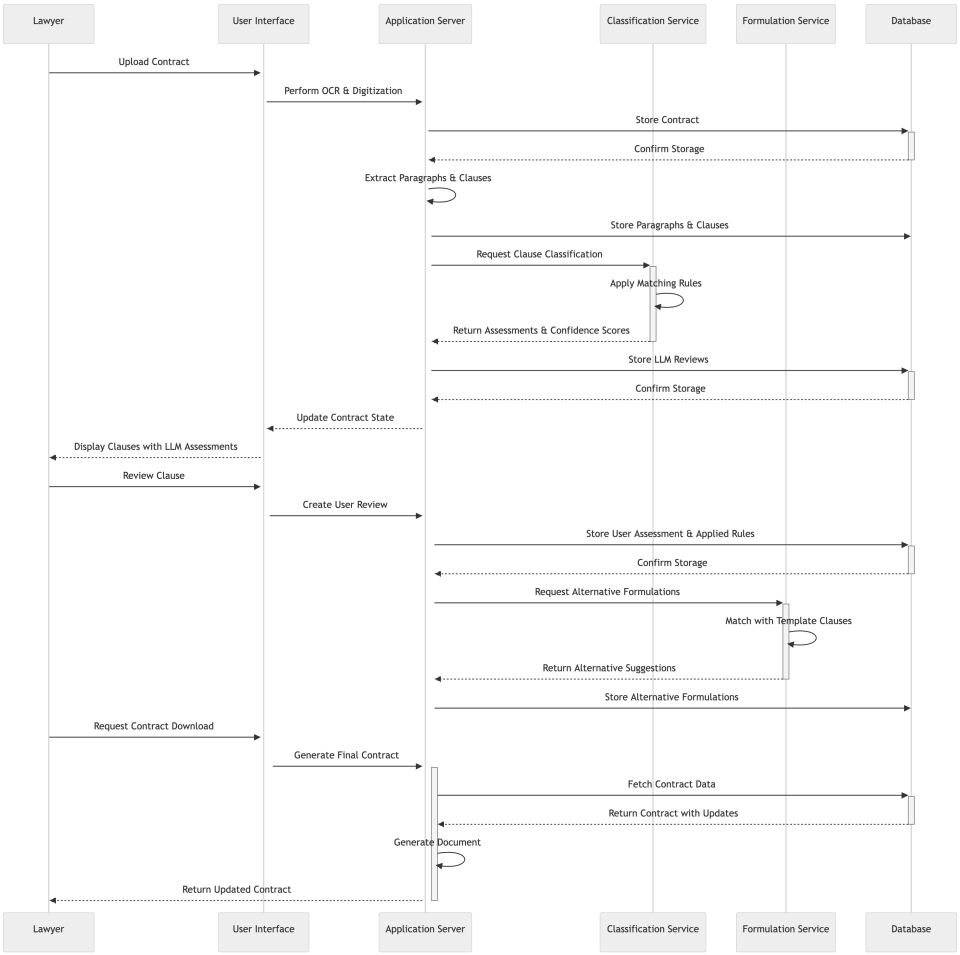


Figure 2.2.: Current workflow of KIBeKodA

3. Related Work

This chapter covers our research within the broader Contract Lifecycle Management (CLM) domain in legal AI. Among different players, Robin AI and Libra Technology were closer reviewed due to their public accessibility. Both systems display different methodologies, from large-scale fine-tuning to customizable AI assistants. We will highlight their features as well as discuss the contrast to the KIBeKodA system.

3.1. Robin AI

One of the notable players in the legal AI space is Robin AI, which fine-tuned its model using over 100 million contract clauses - a fundamentally different approach than the approach employed by the KIBeKodA system [18]. Nevertheless, this highlights Robin AI's system can offer qualitative contract review. It uses a chat-based interface, allowing users to interact with the system to perform tasks such as:

- Defined Terms: Users can specify preferred terminology, such as "client" versus "customer".
- Playbooks: The system suggests tailored contract improvements, enhancing wording and security.
- Find: It enables users to locate specific sections within contracts, such as sections about "Governmental laws" or "Parties".
- Summarize: The AI summarizes contracts by highlighting essential clauses.

Additionally, Robin AI offers a Word Add-On that integrates its Large Language Model advancements directly into Microsoft Word, letting users draft and review contracts within the software.

3.2. Libra Technology

Libra Technology, a legal tech startup based in Berlin, also offers a comprehensive platform with different features. Libra mainly provides a chatbot, where users can select between five different models. In addition to the general-purpose models like ChatGPT 4o and o1, Llama 3.3, and Anthropic's Sonnet 3.5, Libra provides Noxtua 2.6 - a fine-tuned model tailored explicitly for legal applications [19].

Key platform features are:

- Assistants: Select from predefined assistants specializing in different topics or create a custom assistant by uploading documents to build a personalized knowledge base.
- Documents: Upload and securely store documents directly on the platform.
- Prompts: Access a prompt library to ensure more consistent outputs from the LLM.
- Reviews: Add tailored questions to the document analysis for more focused insights.
- Workflows: Design automated workflows tailored to specific legal tasks and processes.

These features, paired with a user-friendly and intuitive UI, make Libra a powerful tool for legal professionals seeking efficient and customizable assistance.

3.3. Position of KIBeKodA

Both Robin AI and Libra Technology provide useful tools for contract management, offering innovative features that enhance the contract lifecycle process. However, KIBeKodA distinguishes itself through an essential aspect: the adherence to current examination guidelines. This ensures that contracts remain legally compliant and reflect recent law changes. While Robin AI relies on fine-tuned models, it cannot consistently integrate the most up-to-date legal updates. Libra Technology offers a partial workaround through its customizable assistants, allowing users to upload documents that contain examination guidelines as a knowledge base.

4. Objectives and Methodologies

This chapter will give the reader a brief overview of the objectives and methodologies used for each research question.

4.1. Objectives

In order to achieve the primary objective of this thesis, three research questions were formulated to facilitate the advancement of the software. Table 4.1 outlines these questions.

ID	Research Question
RQ1	How can changes in examination guidelines and their impact on the evaluation of contract clauses in an in-context learning AI system be tracked and assessed?
RQ2	What is needed to create a user-friendly interface which allows to monitor this evolution without technical knowledge?
RQ3	How can technical aspects and the usability of the system be evaluated?

Table 4.1.: Research Questions

According to these three, we have outlined four major steps, which are illustrated in the figure 4.1.

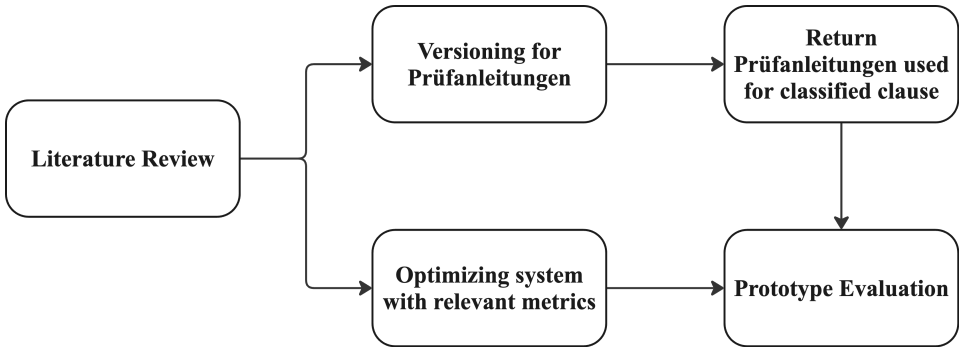


Figure 4.1.: Major steps of research questions

4.2. Literature Review

In this section, we explain the methods used to find relevant literature for this thesis. Our primary search tools included Google Scholar and Google, supplemented by Science Gate to locate applicable papers and articles.

4.2.1. Understanding the System

The initial phase involved gathering information on potential technologies and methods applicable to the legal tech domain. The following queries were conducted:

- "in-context-learning" AND ("law" OR "legal tech"): This query returned 12 results, of which 1 was deemed relevant.
- legal ai evolution: This search produced 4 results, including 1 relevant article.
- "contract lifecycle management": This query returned 4 results, with 2 relevant articles identified.
- "contract lifecycle management" tools: This search yielded only 1 result, which was also relevant.

These queries provided insights about how KIBeKodA works and about how competitors manage Contract Lifecycle Management (CLM). Articles were selected based on their titles and abstracts. With the help of those results, we were able to build a fundamental understanding of what is needed to improve the software.

4.2.2. RQ1: Versioning a Database

A crucial aspect of RQ1 involved understanding how to version a database collection. We conducted the following searches:

- database version control: This query returned 13 results, from which we identified 2 relevant articles.
- "slowly changing dimensions": This search yielded 4 results, with 1 relevant article found.
- "RAG" OR "Retrieval-Augmented Generation": This search produced 16 results.
- prompting AND "in-context-learning": This query returned 4 results with 2 relevant article.
- "chain-of-thought" AND "in-context-learning": This search produced 7 results with 1 being relevant.
- llm prompt structured output: This query resulted in 5 results with 1 being relevant.

This phase helped identify approaches for managing and versioning examination guidelines, focusing specifically on Slowly Changing Dimensions (SCD) types.

Since the current KIBeKodA system did not track or return the specific Prüfanleitungen used in clause classification, we investigated mechanisms for retrieving and displaying these instructions alongside classification outputs. One approach considered was Retrieval-Augmented Generation (RAG), which uses semantic search to return the most relevant results based on similarity thresholds. However, due to concerns about the potential for too many equivalent results that may not be relevant, we chose to adapt our prompt structure within the existing framework.

To refine the prompt structure, we explored various prompting techniques, including Chain of Thought prompting and structured outputs, which could improve response accuracy and consistency. This exploration led to the selection of two specific strategies for prompt adjustment, which we investigated further to determine the most effective approach.

4.2.3. RQ2: Identifying Metrics

To assist users, primarily lawyers, in optimizing the KIBeKodA system without requiring technical expertise, we aimed to identify performance metrics that would be insightful and easily interpretable. Recognizing the importance of a user-friendly interface, we prioritized metrics that could provide immediate, meaningful feedback about the system's functionality and performance.

- "performance metrics" OR "software kpis": This query returned 7 results, with 3 relevant articles identified.
- benchmarking ai systems: Produced 9 results, of which 1 was relevant.
- software development metrics: Returned 12 results, with 2 examined further.
- lawyers performance metrics: This search yielded 4 results, with 1 relevant.
- development metric charts: Produced 26 results, with 7 relevant articles identified.
- react library charts: Returned 3 relevant articles.

Across these sources, it became clear that well-defined metrics and benchmarks are crucial for effective system monitoring and optimization. Given the legal audience, we also focused on visual clarity and simplicity to ensure that insights are immediately understandable.

4.2.4. RQ3: Evaluate the System

We sought methods to evaluate the KIBeKodA prototype for future research. To identify feasible and informative evaluation techniques, we conducted the following searches:

- evaluation software project: This query produced 6 results of which 2 were further investigated

- usability tests: Returned 5 of which 2 were further reviewed

Each of these methods were chosen to provide both quantitative and qualitative insights into system performance and user satisfaction. Usability tests emerged as particularly relevant, as they allow us to gather feedback from lawyers in a structured but flexible manner, ensuring that the system aligns with user expectations and usability standards.

4.2.5. Conclusion of Literature Review

Through these systematic queries and analyses, we established a foundation of relevant technologies, methods, and metrics essential for enhancing the KIBeKodA system. The insights gained have guided our understanding of user-centered design and evaluation methods within the legal tech landscape, ensuring that our approach is aligned with the specific needs and expertise of legal professionals.

4.3. RQ1: Establishing a Version History

Following the literature review, we established a foundation for implementing version control in the KIBeKodA system, focusing specifically on the versioning of examination guidelines. Based on best practices in database management, we examined different approaches within Slowly Changing Dimensions (SCD) [20, 21].

4.3.1. Types of Slowly Changing Dimensions (SCD)

SCD is a methodology used to track changes in database entries over time. It offers three primary types of versioning, each suited to different update requirements [21]:

- Type 1: Involves updating and overwriting an entry if it already exists in the database or creating a new entry if it does not. This process, commonly called "upserting" (update and insert), is straightforward but does not retain historical data.
- Type 2: Preserves historical data by adding a versioning flag to each entry. Each entry is labeled to indicate whether it represents the current version, typically through a field such as `isCurrentVersion`. When a new version of an examination guideline is added, the existing record's flag is updated to `isCurrentVersion = false`, and the new record is created with `isCurrentVersion = true`. This approach enables straightforward retrieval of both current and historical versions. (See Table 4.2 and 4.3)
- Type 3: Is best suited for changes that occur infrequently. Here, the previous and current states are contained in the same row. The database includes fields like `previousText` and `currentText`, where `previousText` holds the older version and `currentText` holds the updated version. For initial entries, `previousText` is set to `NULL`, and only `currentText` is populated. If the examination guideline is revised, `previousText` is updated with the existing `currentText`, and `currentText` is then filled with the new information. (See Table 4.4 and 4.5)

RuleID	Topic	Text	isCurrentVersion
2314	Vergütung, Überstunden	Der Brutto Mindestlohn beträgt 9,82€.	True
2315	Vergütung, Überstunden	Betriebliche Schulungen sind grundsätzlich zu entlohnen.	True

Table 4.2.: Type 2: Table before the change

As shown in Table 4.2, the initial state of the system had two entries.

RuleID	Topic	Text	isCurrentVersion
2314	Vergütung, Überstunden	Der Brutto Mindestlohn beträgt 9.82€.	False
2315	Vergütung, Überstunden	Betriebliche Schulungen sind grundsätzlich zu entlohnen.	True
2314	Vergütung, Überstunden	Der Brutto Mindestlohn beträgt 10.45€.	True

Table 4.3.: Type 2: Table after the change

After the change, a new version of `RuleID` 2314 was added, marking the previous version as outdated (Table 4.3).

RuleID	Topic	currentText	previousText
2314	Vergütung, Überstunden	Der Brutto Mindestlohn beträgt 9.82€.	NULL

Table 4.4.: Type 3: Table before the change

RuleID	Topic	currentText	previousText
2314	Vergütung, Überstunden	Der Brutto Mindestlohn beträgt 10.45€.	Der Brutto Mindestlohn beträgt 9.82€.

Table 4.5.: Type 3: Table after the change

As seen in Table 4.4 & 4.5 the `previousText` changes and receives a value. `currentText` gets updated. No new row is created.

Even though examination guidelines do not change often, they may undergo multiple updates. Also, we have to think about what happens if we decide to give it a new topic as well. Additionally, some updates might involve changing the topic or scope of the examination guideline (e.g., from "Vergütung, Überstunden" to a more specific "Vergütung"). Given this, SCD Type 3 would not meet our needs, as it is best suited for single, infrequent updates. We need a method that can track the changes in each version without deleting old data.

Literature on database management and versioning strategies highlighted the benefits of SCD Type 2 for applications requiring historical data retention and easy retrieval of current and past versions [20, 21].

4.3.2. Return Examination Guidelines used for Classification Clauses

Now that we know how to create a history of examination guidelines, the next step is to identify which specific examination guidelines were used to classify a contract clause. Currently, the KIBeKodA system's workflow involves sending all available examination guidelines along with a single contract clause to the LLM, such as OpenAI's, which operates as a "black box". The LLM processes the clause and returns an assessment - classifying it as either "valid", "unfair", or "invalid". However, there is no explicit link between the assessment and the specific examination guidelines applied in making that decision.

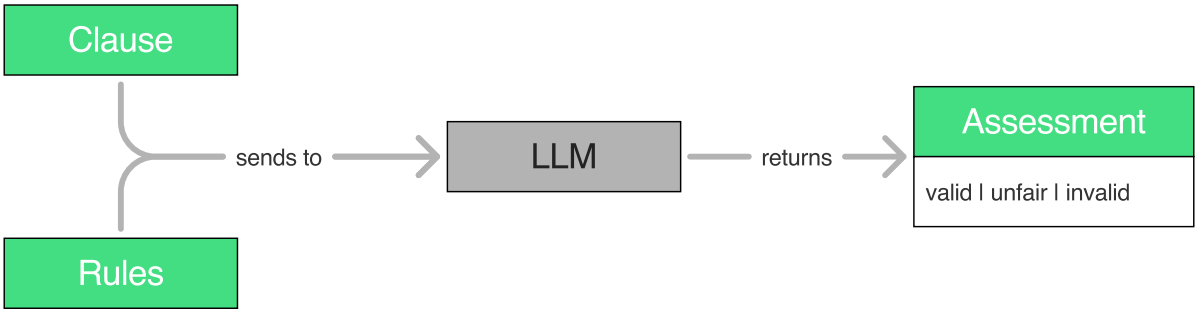


Figure 4.2.: Evaluation process before applying methods

The KIBeKodA system includes a classification service implemented in Python, which performs the following steps. The contract clause and all examination guidelines are combined into a structured prompt instructing the LLM on evaluating the clause, using the rules as guidance. The prompt specifies both the evaluation process and the desired structure of the output. The prompt is then sent to the LLM, which returns an assessment without direct reference to specific examination guidelines. The classification service then filters the response and finds the correct assessment. We will investigate on prompt engineering and how it could enable the KIBeKodA system to provide both an assessment and a list of the examination guidelines applied. Specifically, we will elaborate on the CO-STAR method [22], which structures prompts to provoke more precise, specific LLM' responses. CO-STAR stands for:

- **C:** Context of the prompt

- **O:** Objective, the goal of the prompt
- **S:** Style, the desired style of writing in the answer
- **T:** Tone, to ensure the output is phrased with the correct sentiment
- **A:** Audience, the response is written for lawyers
- **R:** Response, we can establish a desired format

This method provides a structured approach to crafting prompts that produce more precise and valuable responses from LLMs. Additionally, we will investigate on OpenAI's structured output, which enables to create a predefined schema that the answer shall follow, specifying fields for both the assessment (valid, unfair, or invalid) and a list of the specific examination guidelines used in reaching that assessment. This approach is more flexible than traditional `json_mode` output, as it creates a consistent, machine-readable format that associates each assessment with the relevant examination guidelines [23].

4.4. RQ2: Optimizing the System with Relevant Metrics

This section outlines the methodologies used to identify and implement key metrics that enable legal professionals to monitor and optimize the KIBeKodA system without requiring technical expertise. Using insights from the literature review, we adapted traditional software development metrics to meet the specific needs of the system, categorizing them as follows:

- System Information Metrics: These metrics provide a snapshot of the system's current state without comparing it to prior states
- Detail Information Metrics: These metrics enable users to compare the system's performance across different versions and timestamps, including updates to examination guidelines and classification results

Below, we describe each selected metric and its relevance to our system, drawing analogies to established software development metrics where applicable.

4.4.1. Confidence Score for Examination Guidelines

The confidence score, in general, describes the confidence of the LLM when it classifies a clause in the contract. For this metric, we want to measure how well it can apply the examination guidelines and how certain it is in doing so. According to recent research, we can "just ask for calibration" [24] to produce well-calibrated confidence scores from language models. These verbalized confidence scores - explicitly expressed as probabilities or linguistically - are often well calibrated. Ke Shen and Mayank Kejriwal's research suggests to introduce a risk-adjusted calibration method that further refines confidence scores by addressing "decision and composite risks" [25]. This approach, known as "Deciding when to Decide", helps LLMs to navigate through uncertain inference tasks by adjusting raw confidence scores to better reflect true certainty levels.

4.4.2. Time Spent per Contract (LLM only)

This metric tracks the LLM's processing time for each contract, normalized to an average per 100 words. By excluding lawyer involvement, it ensures that lawyers are not pressured to increase speed at the expense of accuracy. Time comparisons between different LLM models can reveal efficiency gains. This is similar to the "cycle time" metric in software development, which measures the time for an entire process step until the user, i.e., lawyer, receives output. Another metric, lead time, could not be used since we don't measure the time from uploading the contract until the contract is evaluated by the lawyer [26].

4.4.3. Confusion Matrix and Accuracy Score

The Confusion Matrix is a standard metric in software development, especially in machine learning and AI applications. It helps evaluate the LLM's performance by comparing its output with ground truth classifications provided by lawyers [27]. The classical confusion matrix is structured as follows:

- True Positive (TP): Both the lawyer and LLM classify a clause as valid
- True Negative (TN): Both classify a clause as either unfair or invalid
- False Positive (FP): The LLM classifies a clause as valid, but the lawyer classifies it as unfair or invalid
- False Negative (FN): The LLM classifies a clause as unfair or invalid, but the lawyer classifies it as valid

Since the above confusion matrix is only 2x2, we adapted this metric to accommodate specific classifications used in our system: valid, unfair, and invalid.

LLM \Lawyer	**Valid**	**Unfair**	**Invalid**
Valid			
Unfair			
Invalid			

Table 4.6.: 3x3 Confusion Matrix comparing LLM and lawyer classifications

Derived from the confusion matrix, the accuracy score shows the percentage of correct classifications by the LLM. It is calculated by accumulating the diagonal of the matrix providing a straightforward indicator of the system's performance. Also, filtering for specific examination guidelines helps to identify good and bad performing rules.

This overview of some of the metrics used in the system helps the user to understand the next chapter, 5, better.

4.5. RQ3: Prototype Evaluation

This section outlines the methods used to evaluate the KIBeKodA prototype. The evaluation combines technical and usability aspects to assess system functionality and user experience.

The chosen evaluation method is an unmoderated usability session. While such sessions are less controlled and may lack context or result in less accurate findings [28, 29], they indicate that the success of online usability surveys strongly correlates with participants' familiarity with computers [30]. Given the technical proficiency of the lawyers involved, this approach seems suitable for collecting meaningful feedback on the system.

Real-world scenarios are designed to reflect the practical challenges lawyers face, enabling the assessment of system performance and the overall user experience. The evaluation process focuses on two key areas: system functionality and usability.

4.5.1. System Functionalities

To assess the technical aspects, the evaluation targets the following core functionalities:

- Contract Analysis: Verify that the system applies examination guidelines correctly and displays them with clear, helpful AI comments.
- Versioning of examination guidelines: Ensure that the system recognizes the edited examination guideline and accurately distinguishes between semantic and non-semantic changes.
- Metrics: Assess whether metrics are fetching and displaying data user-friendly and understandable.
- Impact of Rule Changes: Evaluate the system's ability to detect semantic changes in applied examination guidelines, flag impacted clauses with the `toreassess` attribute, and allow lawyers to reevaluate them using AI.

One task involves testing an examination guideline, simulating real-world updates by introducing both semantic and non-semantic changes to the rule, and observing the behavior of the system.

4.5.2. Usability Evaluation

To complement the technical evaluation, participants provide qualitative feedback on the system's design, usability, and intuitiveness. Lawyers are asked to perform specific tasks within the system, such as navigating the contract analysis and analytics pages, and are then prompted to answer a set of structured and open-ended questions to measure the user's subjective assessments of the system [28].

The usability assessment addresses the following areas:

- Ease of Use: Lawyers rate their experience with tasks such as conducting a contract analysis, editing examination guidelines, and interpreting analytics metrics.

- System Navigation: Participants evaluate the clarity and intuitiveness of the system's interface.
- Perceived Complexity: Questions focus on whether any aspects of the system feel unnecessarily complex.
- Suggestions for Improvement: Open-ended questions allow participants to provide detailed feedback on potential enhancements, design changes, or errors they faced.

4.5.3. Questionnaire Design

The questionnaire is designed to provide both structured and qualitative insights. It includes three main categories:

- Personal Background: Information about professional experience and familiarity with the system.
- Task-Specific Usability Questions: Likert-scale questions to estimate the participants' experiences with specific features, such as the contract analysis process and analytics page.
- Open-Ended Feedback: Questions inviting detailed observations and suggestions for improvement, including potential future features and feedback on design and performance.

For usability-related questions, the evaluation utilizes elements of the System Usability Scale (SUS) [31]. It provides a robust tool for measuring usability, offering a standardized approach to assess participants' experiences with the system. Alongside custom questions tailored to the unique features of KIBeKodA. A detailed view of the questionnaire can be found in Table A.2.

4.5.4. Data Analysis

Quantitative data, such as Likert-scale responses and task completion rates, are analyzed to assess system performance and usability. Qualitative feedback from open-ended questions provides deeper insights into areas for improvement and user satisfaction.

By combining functionality testing with usability feedback, this evaluation design provides a comprehensive overview of the system's strengths and limitations, forming the basis for future refinements.

5. Results

This chapter addresses the research questions as seen in Table 4.1. It explains how we implemented the elaborated methods from the literature review.

5.1. Save and Store Versions of Examination Guidelines

Our existing system initially employed Slowly Changing Dimensions (SCD) Type 1, where old data is overwritten. However, based on findings from our literature review [20, 21], we transitioned to SCD Type 2. This approach allows us to maintain historical versions examination guidelines, which is crucial for tracking changes over time. By using SCD Type 2, we ensure that the current version of a rule is flagged with `isCurrentVersion = true`, which ensures that only the latest version is used during clause evaluation. Also, historical versions remain intact, allowing us to compare previous versions with the current one, thereby enabling a thorough assessment of how changes in guidelines affect contract clause evaluations.

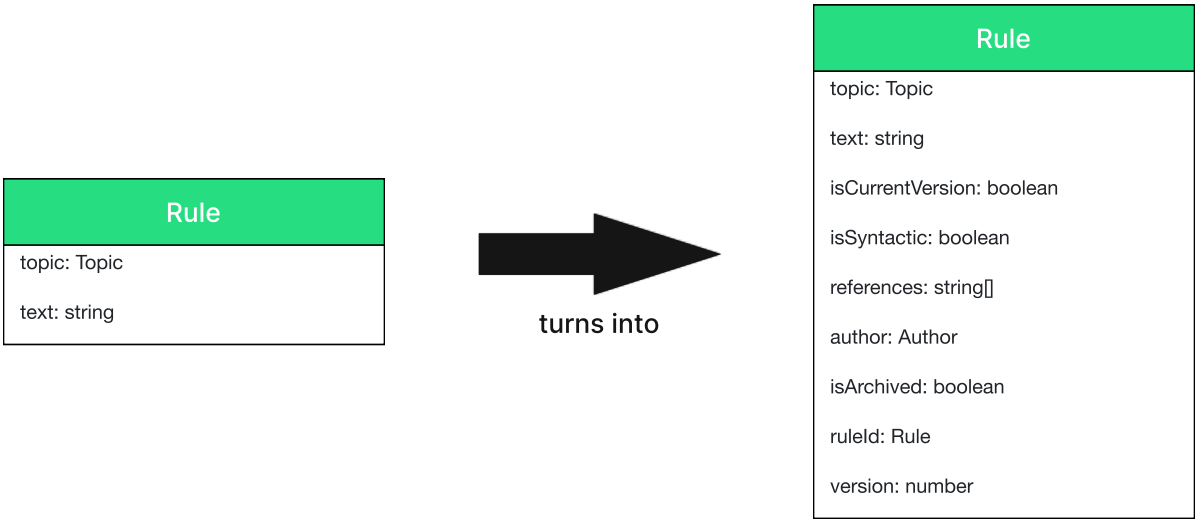


Figure 5.1.: Before (left) and after (right)

In Figure 5.1 you can see the updated rule class which supports versioning. Several adjustments were made to the schema. First, we introduced new attributes such as `isSyntactic`, a boolean flag that indicates whether a change is just syntactic or semantic. This distinction helps to avoid unnecessary creation of new versions when only minor syntactic changes are made. Additionally, we added a `ruleId` attribute to link different `versions` of the same

examination guideline together, ensuring that all related versions can be easily retrieved and compared.

Instead of deleting outdated rules, we now allow users to archive them. This approach ensures data consistency while maintaining access to historical records for future reference. `references` for rules are another feature that will be explored further in future work.

5.2. Tracking Changes of Examination Guidelines

The transition to SCD Type 2 allows us to track changes in examination guidelines more effectively by maintaining a complete history of all versions. The system provides functions such as `getRuleHistory(ruleId: string)`, which retrieves all versions of an examination guideline and sorts them by date. This allows us to analyze how changes over time might influence contract clause evaluations. The distinction between syntactic and semantic changes plays an important role in this tracking process. By introducing the `isSyntactic` attribute, we can differentiate between updates that are purely syntactic - therefore do not require a new version - and those that involve significant semantic changes. This distinction helps ensure that only significant updates trigger new versions. Semantic changes are likely to have a direct impact on how contract clauses are interpreted by the AI system. By comparing old and new versions of examination guidelines, we can analyze shifts in interpretation that may arise from these changes. Syntactic changes on the other hand are only small adjustments that should not influence the LLMs evaluation. `getAllCurrentRules(isArchived?: boolean)` returns all current rules, optionally also the archived ones.

5.3. Prompt Modification

In response to the findings from the literature review and the need for a more structured output, we made significant adjustments to the prompt used in the KIBeKodA system. This need for more transparency is notable.

The previous prompt, while functional, lacked certain elements that would ensure a more structured and consistent output when evaluating contract clauses. To be maintainable by legal professionals without technical expertise, it is crucial that each classification clearly indicates the specific examination guidelines applied, allowing lawyers to quickly understand and validate the results. Below, we will discuss the key changes made to the prompt and their impact on improving the system's performance.

One of the most important changes we introduced was the use of a structured output format. This allows a more organized and reliable data generation, as it follows a specific format that can be easily parsed and used for further processing [32]. Previously, the system's output was not well-structured, making it difficult to consistently interpret or process the results. To address this, we implemented OpenAI's structured output option, which allows us to define a schema for how we want the model's responses to be formatted. The new schema ensures that each evaluation includes essential information such as the `clauseId`,

`assessment` (whether the clause is void, unfair, or valid), relevant `ruleIds`, `confidence` level, and a `comment` justifying the decision. The schema is defined as follows:

```
class ClauseValidationExpert(BaseModel):
    clauseId: str
    assessment: Literal["void", "unfair", "valid"]
    ruleIds: List[str]
    confidence: float
    comment: str
```

This change alone does not guarantee that the output will adhere to this structure. Therefore, we also adjusted how we formulate the prompt to guide the model toward producing responses that fit this schema. With these modifications, we also potentially influence the LLM responses and change the results to earlier implementations.

The previous version already used some aspects of prompt engineering and followed a Chain of Thought (CoT) approach. CoT prompting helps the LLM to break "down a complex problem into smaller steps", which "helps the LLM to understand the problem more deeply, and to generate more accurate and informative answers" [33]. However, this approach was not fully optimized for generating structured outputs. To enhance this process, we adopted the CO-STAR approach (Context, Objective, Steps, Task, Action, Result), which was created by Sheila Teo, winning her a prize at the GovSingapore GPT-4 Prompt Engineering challenge [22]. This approach helps guide complex reasoning tasks in a structured manner [34, 35].

In our revised prompt design, we provided clear instructions to ensure that the model returned responses in a structured format. The user prompt now explicitly asks for an evaluation using specific rules and instructs the model to return its response in a predefined structure. Best practices in prompt engineering emphasize the importance of being clear and specific when crafting prompts. For example, providing context and specifying desired output formats are essential for ensuring that LLMs generate accurate and relevant responses [33].

The refinements made to the prompt in the KIBeKodA system are important for the following improvements. The biggest issue with the new changes, we were able to resolve was the occasional return of all rules when no rule was applicable. This problem, although infrequent, created inconsistencies in the system's output. To address this, we implemented additional backend logic that ensures more reliable and consistent results, allowing correct output also when there are only a few rules added to the system.

By implementing best practices from the literature, we have improved the structure and transparency of the LLMs evaluation on the clauses and providing reliable and consistent outputs.

5.4. Managing Rule Updates and Clause Reassessment

The whole setup helps to identify clauses that are potentially not up to date if an examination guideline changes due to law text changes or semantic errors in one of the rules. We can trace back which clause and which review is affected by those adjustments. Once a rule

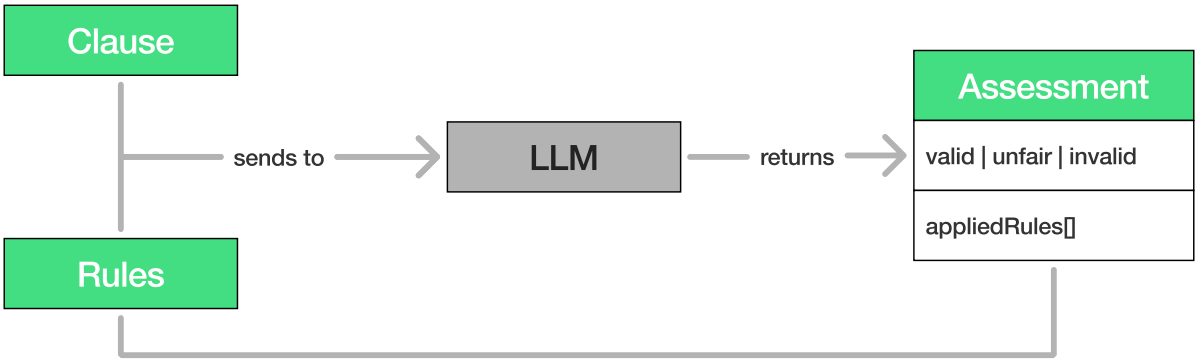


Figure 5.2.: Evaluation process after applying methods

changes semantically, the system searches for reviews, where the examination guideline was applied and changes the `assessment` field to `toreassess`. This helps lawyers to find clauses that might need a reevaluation.

5.5. Combination of LLM Assessment and Lawyer Review

In the KIBeKodA system, we initially separated the assessments generated by the LLM from those provided by human lawyers. However, with recent improvements to the structure of the prompt and the evaluation process, we found that the LLM's output became more similar to a lawyer's review. This led us to combine both types of reviews under a unified structure in the backend, which not only simplified the system but also extended its functionality.

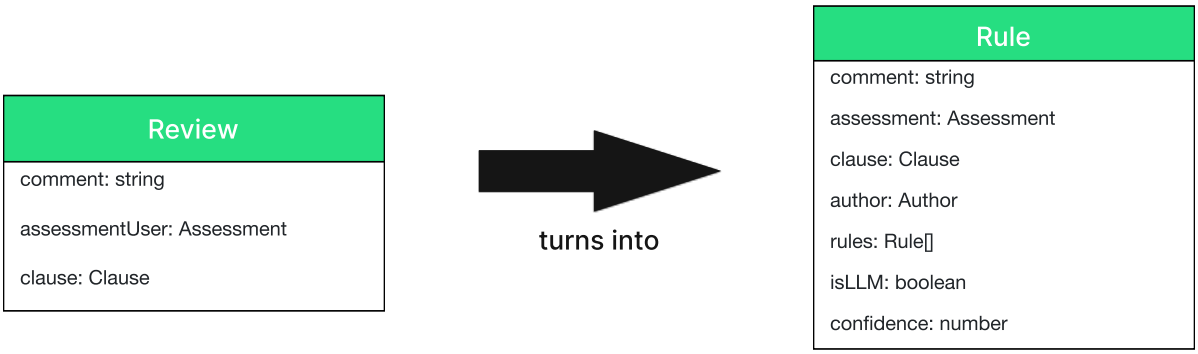


Figure 5.3.: Before (left) and after (right)

The core of this change was the modification of the Review class. Previously, the class consisted of three main attributes: `comment`, `assessmentUser`, and clause. The `comment` field stored any remarks made by the lawyer, while `assessmentUser` held the assessment provided by the lawyer for a given contract clause. The `clause` attribute linked each review to a specific contract clause. In order to keep both the LLM and lawyer review under one structure we introduced several changes. First, we renamed the `assessmentUser` to a more general

assessment attribute. Next, we added an `author` attribute to represent who created the review. The Author class is an abstract model that can represent either a user (lawyer) or an LLM. Both share common attributes but are differentiated where necessary for specific tasks. This distinction is important for certain logic in the backend, where different handling may be required depending on whether a review was generated by a machine or a human. Another added attribute is `rules`, which stores an array of `ruleIDs` that were applied during the LLMs evaluation process. This field especially is important for machine generated classifications because it tracks which rules were used. For user reviews this field is left empty. Additionally we introduced a fourth state: `toreassess`, which indicates whether any of the rules in the rules array were changed after being applied. When set, it implies that an applied rule was semantically adjusted and the review might have to be reevaluated. Finally, we added a `confidence` field particularly for the AI reviews which display the `confidence` of the evaluation.

5.6. Additional Implementations

This section outlines noteworthy features that were not directly part of the thesis objectives but proved valuable in understanding the system or contribute to facilitating future research.

5.6.1. Adding User Roles

In the current KIBeKodA system, all users have full and unrestricted access to all functionalities. This includes that a user (e.g. student) has the ability to delete another user (e.g. lawyer), thereby denying the lawyers access to the system. This is undesirable behavior. To address this issue we introduced two user roles, with the potential to add more in the future:

- Admin: has full access to the entire system
- User: has restricted access to specific areas

The following code snippet demonstrates how permission control is implemented in the updated system:

```
@Route('users')
...
@Get('')
@Security('jwt', [AuthorRole.ADMIN])
@Middlewares(requireRoles([AuthorRole.ADMIN]))
public async getAllUsers(): Promise<UserResponse[]> {
    ...
}
```

In this example, we are working with the route `users/`, where access is restricted based on user roles. The decorators `@Security` and `@Middlewares` are used to enforce these restrictions. `@Security("jwt", [AuthorRole.ADMIN])` ensures that only users with the admin role can

access this route. It uses JSON Web Token (JWT) for authentication, which is a best practice for stateless authentication and "one of the most used authentication standards in web applications" [36]. JWT uses robust signing algorithms to prevent forgery and ensure security. Once a user signs in, an expiration time is set on the token, allowing access without signing in again until the token expires [37]. `@Middlewares(requireRoles([AuthorRole.ADMIN]))` checks whether the current user has the necessary role (i.e., admin) to access this route. `@Middleware` acts as an intermediary between the operating system and applications, facilitating communication and data management in distributed systems [38]. In this case, it ensures that only authorized users can access certain functionalities within the system. Both `@Security` and `@Middleware` can be applied at either the macro level, restricting an entire route like `users/*`, or at the micro level, restricting specific paths like `users/`.

5.6.2. Migration to MongoDB Atlas

As illustrated in Figure 2.1, the KIBeKodA system initially utilized a formulation service to retrieve template clauses that were semantically similar to clauses evaluated as `unfair` or `invalid`. This service relied on ChromaDB, a vector database designed for handling vector-based searches, which became necessary due to MongoDB's lack of native support for such queries at the time. Unlike traditional keyword-based queries, which depend on exact matches, vector searches leverage vector embeddings to represent the semantic meaning of data, enabling more nuanced and context-aware retrieval [39]. However, maintaining two persistent databases - MongoDB for general data storage and ChromaDB for vector searches - introduced unnecessary complexity and increased maintenance overhead. Given that all other system data, such as contracts and user information, was already stored in MongoDB, we decided to consolidate the architecture by removing ChromaDB. This decision was motivated by MongoDB Atlas's recent introduction of native vector search capabilities, which allowed us to integrate all data-related operations into a single platform.

To simplify the system architecture, we migrated to MongoDB Atlas and incorporated the template clause functionality directly into the application server, effectively rendering the formulation service obsolete. As a result, we were able to remove this component from the system architecture entirely, as depicted in Figure 5.4. This migration not only simplified the architecture but also reduced operational complexity and improved maintainability.

To complete this migration successfully, we transferred all relevant code from the formulation service, originally implemented in Python, into the application server. This required refactoring parts of the codebase and implementing new components within our main component. Specifically, we introduced a new controller, service and model to handle both storage and retrieval of template clauses in MongoDB Atlas. One key challenge during this migration was adapting certain functionalities that were previously handled by ChromaDB's in-house features, such as finding nearest embeddings. To address this, we used OpenAI's embeddings model `text-embedding-ada-002` for generating embeddings when storing and searching template clauses. MongoDB's native vector search feature was then used to efficiently retrieve semantically similar clauses from within the database. This change drastically simplified the systems architecture making it easier to maintain and reducing complexity.

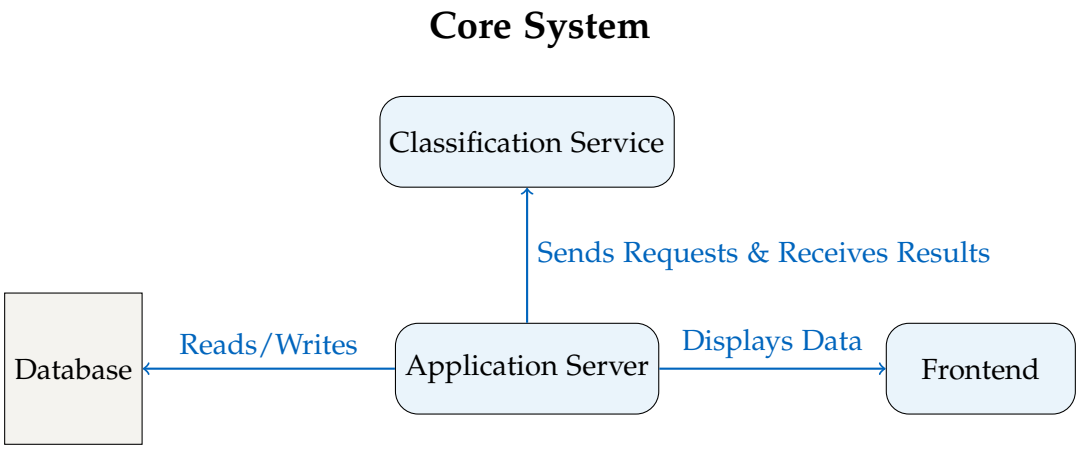


Figure 5.4.: System Architecture of the main components after removing the Formulation Service

5.6.3. Establishing a Topic Class

Currently, each examination guideline and template clause is associated with a topic; however, these topics are neither standardized nor systematically managed. This leads to inconsistencies and can result in time-consuming updates when topics are modified, such as when they are made more granular or consolidated.

To address this issue, we created a dedicated Topic Class and linked it to both examination guidelines and Template Clauses. The structure of this implementation is straightforward, as shown in Figure 5.5.

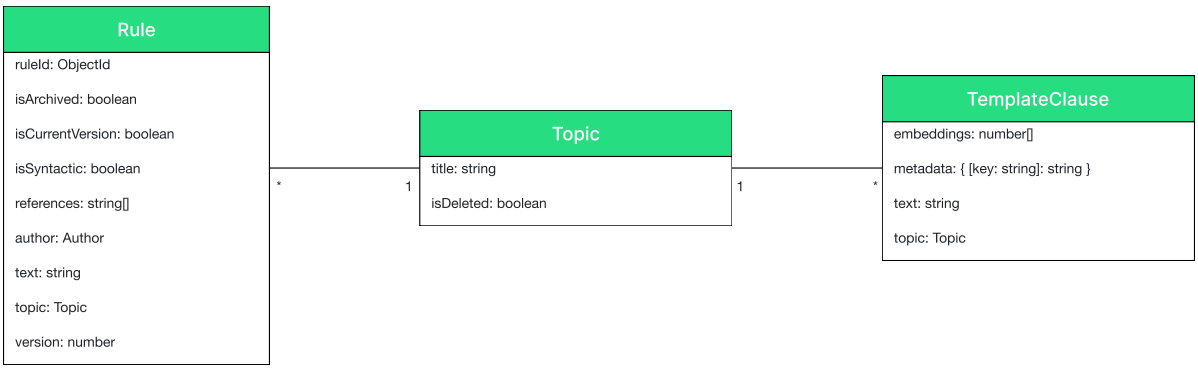


Figure 5.5.: Implementation and interaction with Rule and Template Clause class

This enhancement involved introducing a new model, controller, and service. The Topic model is connected with the Template Clause and Rule models. To maintain data consistency, the system checks if a topic with the same name already exists before creating a new one, ensuring that duplicate topics are avoided.

By implementing the Topic Class, we improve data consistency and provide a more structured foundation for future modifications and expansions.

5.7. Analytics Page

The analytics part is newly introduced with this thesis. First, we had to create a new database schema to save needed information. Since contracts are the main part of the analytics, we save the contracts with all important information once the lawyer marks them as reviewed ("anwaltlich abgenommen").

```
export interface IAnalysis {
    _id: Types.ObjectId;
    contractId: Types.ObjectId;
    contractTitle: string;
    industry: string;
    reviewedAt: Date;
    metrics: IAnalysisMetrics[];
    clauses: {
        clauseId: Types.ObjectId;
        content: string;
        position: number;
        pageNumber: number;
        llmReviews: {
            assessment: string;
            confidence: number;
            appliedRules: {
                ruleId: Types.ObjectId;
                ruleText: string;
                topic: Types.ObjectId;
                version: number;
            }[];
            createdAt: Date;
        }[];
        userReviews: {
            assessment: string;
            createdAt: Date;
        }[];
        createdAt: Date;
        updatedAt: Date;
    }[];
    createdAt: Date;
    updatedAt: Date;
}
```

As seen in the code snippet above, the analytics object is comprehensive. We save basic contract information, metrics for the specific contract, such as contract length and average speed of the LLM. We store the clauses including the LLM and user reviews and also the

applied rules in the object. Additionally we have some meta information such as `createdAt` and `updatedAt`.

In order to access the objects we also introduced a new controller for the analytics, retrieving and creating them once a contract is set to `FINISHED_REVIEW`.

5.8. Frontend Adjustments

This section describes the UI improvements made to several pages in the system.

5.8.1. Rule Overview and Search

Starting with the changes of the examination guidelines, we introduced small design adjustments such as modern and colorful buttons to edit, see versions and archive rules. We further added a search bar to increase the user experience, when the user is looking for a certain rule or topic (See Figure5.6). If an examination guideline has references, the table row acts like an accordion, by clicking on it, the references can be seen.

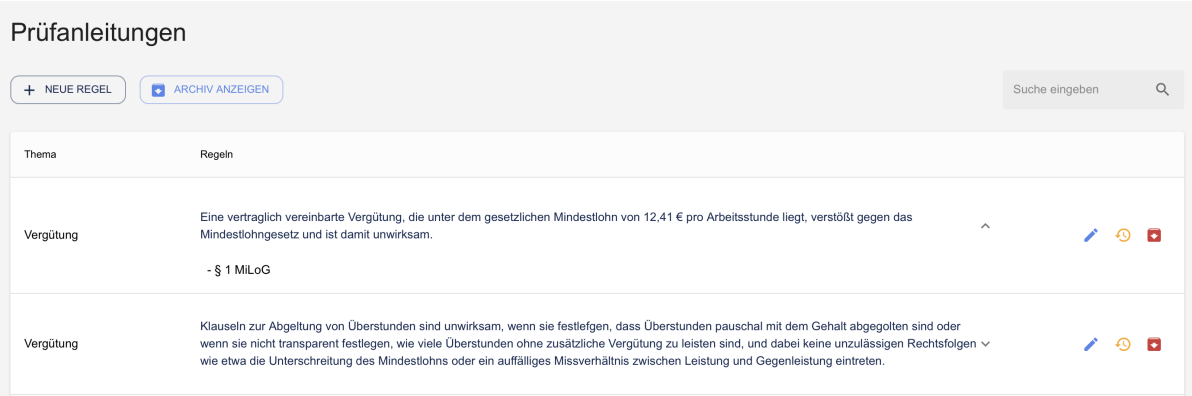


Figure 5.6.: Rule overview with search bar and expanded accordion

5.8.2. Rule Creation and Editing

When the user is editing a rule, the topic can be changed as well as the references and text. If no semantic changes have been made, a checkbox can be clicked "keine semantische Änderung" which then sets the boolean of `isSyntactic` to true. Another user friendly feature is the autocomplete in the modal, when you create or edit an examination guideline. This is a feature of the used design framework Material UI. It takes all topics as options and filters on input change. If there is no such topic, a new one can be created without changing the interface leading to a better user experience.

Figure 5.7.: Topic with autocomplete function

5.8.3. Template Clause Overview and Search

Since the template clause page is very similar to the examination guidelines, but not part of this thesis, we added the same features here, without the version control, as an additional feature. (See Figure B.1)

5.8.4. Topic Management

Another extra feature is the control of topics. Similar to the other two pages, topics can be created and edited. If a topic has no examination guidelines and no Template Clauses, its attribute `isDeleted` can be set to `true`, which means that the topic is not deleted but rather removed from the frontend. This add-on further ensures data consistency and increases user experience. In Figure 5.8 the new page can be seen.

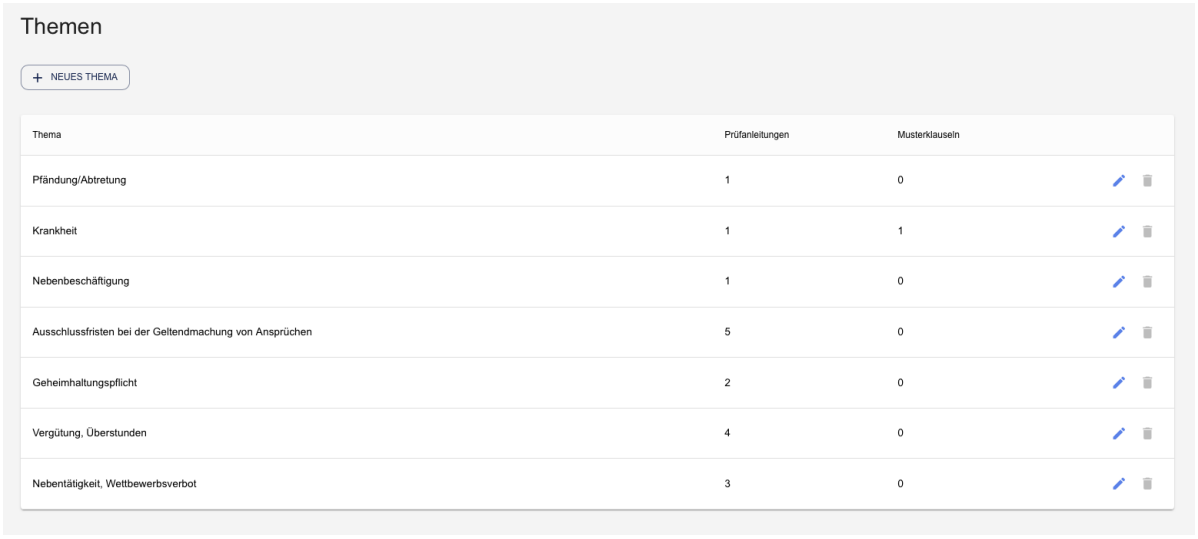


Figure 5.8.: Topic overview

5.8.5. Improved Contract Analysis

The core functionality of the KIBeKodA system revolves around contract analysis, where both LLMs and lawyers evaluate contract clauses. Previously, the system only displayed

the outcome of the evaluation (i.e., whether a clause was "valid", "unfair", or "void") using color-coded indicators. However, it did not display any thoughts and comments of the AI. This limitation made it difficult for users to understand the reasoning behind evaluations, thus increasing time spent on the contract or clause. With the recent refinements in the prompt and backend structure, the system now provides a more detailed insight into the LLMs evaluation process.

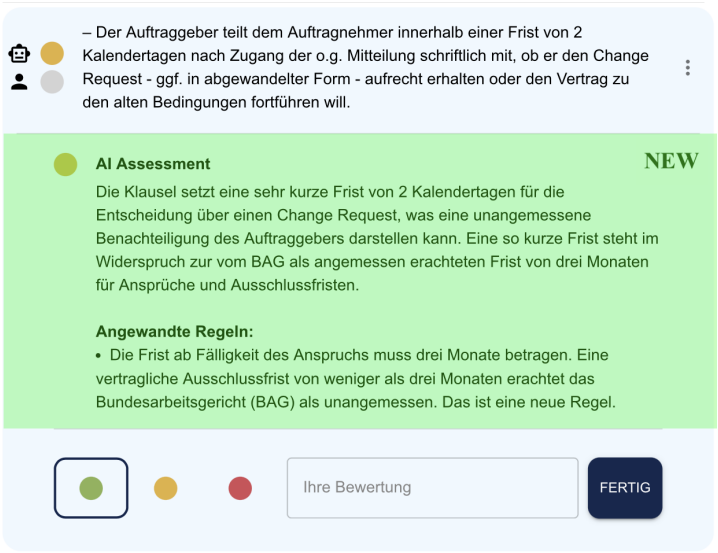


Figure 5.9.: Updated design AI assessment

As shown in Figure 5.9, there is an entirely new section that includes additional information for each assessment. The top part displays the comment along with the author's name. Since the language model's name (e.g., gpt-4o-20-11-24) can be complex, we decided to use a general label, "AI Assessment". This is where the `author` attribute in the `Review` class becomes relevant. In the comment section, the LLM explains how it reached its result (e.g., "unfair"). By making this reasoning visible, lawyers can immediately understand the LLM's decision-making process. Below the comment are the applied rules, including an exact copy of the applied examination guideline. This helps lawyers verify whether the rule was applied correctly by the model, allowing them to more quickly confirm or challenge the decision.

These changes in the contract analysis improve the transparency, user experience and follows the objective of the system, reducing time and lowering costs. Reviews can be quickly validated or challenged, without much effort.

Mistakes can occur, not only by the machine but also by lawyers. To address this, we enhanced the functionality of the reset button. Previously, users could only reset the entire process, which involved restarting OCR and redoing the entire AI evaluation. We introduced a more granular reset functionality. Depending on which step of contract evaluation is currently active, users can now reset to one of the following stages:

- Vertrag neu analysieren: Resets to the beginning; only the uploaded PDF remains

available.

- Alle Klassifizierungen zurücksetzen: Removes all lawyer reviews and redoes the AI classification using data from OCR.
- Anwalt Reviews zurücksetzen: Removes all user reviews and restarts from the lawyer's evaluation process.
- Nicht anwaltlich abgenommen: Marks that legal review is not yet done.
- Nicht abgeschlossen: Indicates that the lawyer's review is still ongoing and has not been finalized.

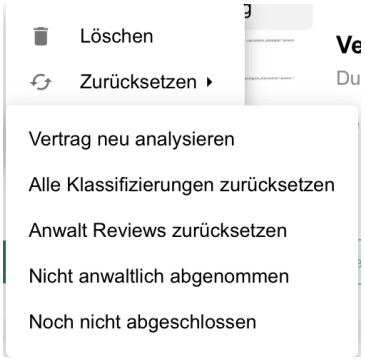


Figure 5.10.: Reset button with reset options

Once the lawyer is done reviewing the contract, a click on "anwaltlich abgenommen" updates the state of the object, which guarantees that the contract has been fully reviewed.

5.8.6. Adding a new Filter

As seen in the Figure 5.11, the system has four filters for the clauses:

- "Unzulässig": invalid clauses that *give* an disadvantage to one contract party
- "Unfair": unfair clauses that *might* give an disadvantage to one contract party
- "In Ordnung": valid clauses that are correct according to legal standards

If you click on one, the clauses in this category are filtered. As mentioned before, due to changes in the backend logic, we now have a fourth state:

- "Zur Neubewertung": to reassess

This state is for all clauses, where a review potentially needs a reevaluation, due to a change in an applied examination guideline. A figure with all adjustments in the header, including filters and reset functionality can be viewed in Figure B.3.

Figure 5.11.: Four options for filtering clauses

5.9. Implementing the Analytics Page

A new part of the system is the analytics page, where we implemented the metrics found in the literature review. To maintain a consistent design across the system, we utilized the ApexCharts library, which offers a wide range of customizable chart designs suitable for various data visualizations [40].

The first step in implementing the analytics page was to create a general structure that allows for proper data handling. This was achieved by introducing a main component that manages the retrieval of data through five queries:

- basicMetaQuery: This query gathers basic information about the system, such as the total number of examination guidelines, users, topics, and contracts, as well as details about the current LLM in use.
- metricsDataQuery: This query retrieves metrics related to LLM performance, such as the time spent per contract.
- analysisDataQuery: This query is used to gather data for calculating confidence scores for the applied rules based on LLM assessments.
- invalidReviewsQuery: This query retrieves all contracts that contain clauses flagged as outdated.
- confusionMatrixQuery: This query is essential for generating the confusion matrix and calculating accuracy scores based on comparisons between LLM assessments and human reviews.

These queries are then further processed to be used in the single metrics, which will be presented in the following subsections.

5.10. System Information

The System Information tab provides users with a comprehensive overview of the system's current state at a glance.

5.10.1. Basic Information

The basic information gives a rough and quick overview of the systems state. It displays the amount of core attributes, such as total amount of contracts in the system, amount of topics, examination guidelines and users which is complemented by the currently used LLM.

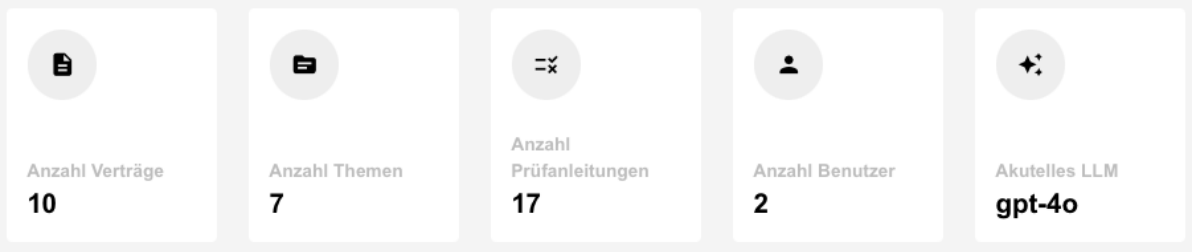


Figure 5.12.: Basic information about the system

5.10.2. Time Spent per Contract (LLM only)

The Time Spent per Contract (LLM only) graph is presented as a box plot. This visualization compares the performance of different LLMs by showing how long each model takes to process contracts. Each box plot consists of three key values (Figure 5.13):

- Min: The fastest processing time recorded for a specific LLM.
- Median: The middle value, showing the median processing time.
- Max: The slowest processing time recorded for that LLM.

To ensure consistency across varying contract lengths, all processing times are normalized to average times per 100 words. Additionally, the graph only includes data from complete evaluations, providing a reliable representation of model performance.

5.10.3. Examination Guidelines over Time

The Number of Examination Guidelines over Time graph (Figure 5.14) uses a line chart to display how the number of rules has evolved over time. This metric reflects the growth or changes in the system's rule set, providing insights into how frequently new examination guidelines are added. Each data point on the chart represents the total number of examination guidelines on a specific date. By hovering over a data point, users can view the exact number of rules at that time, offering a detailed snapshot of the system's progression.

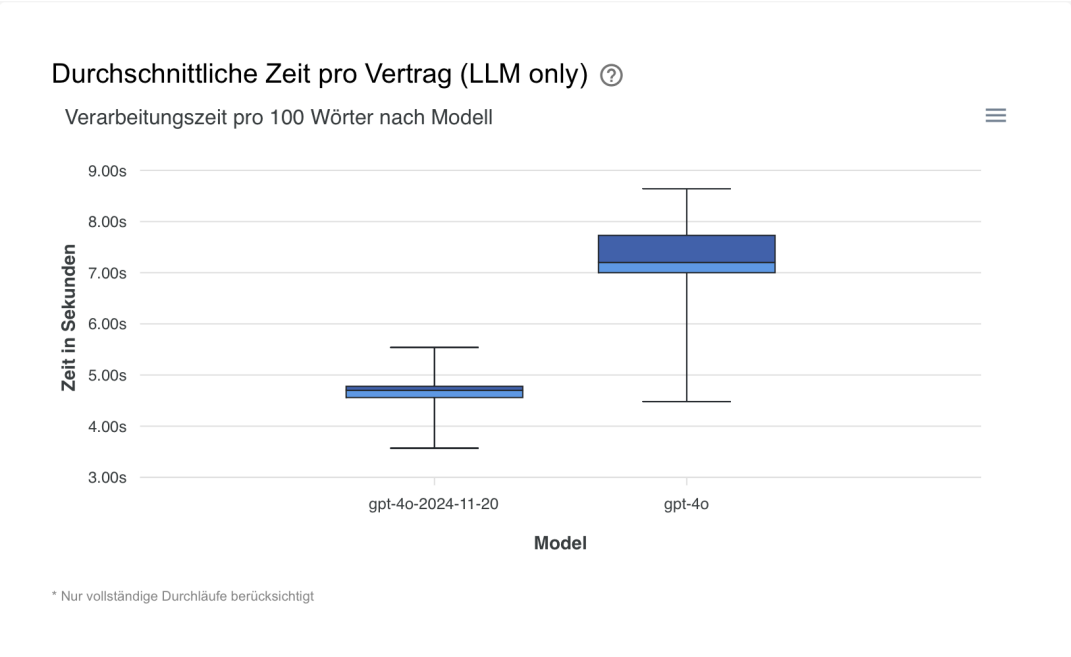


Figure 5.13.: Time spent per contract (LLM only)

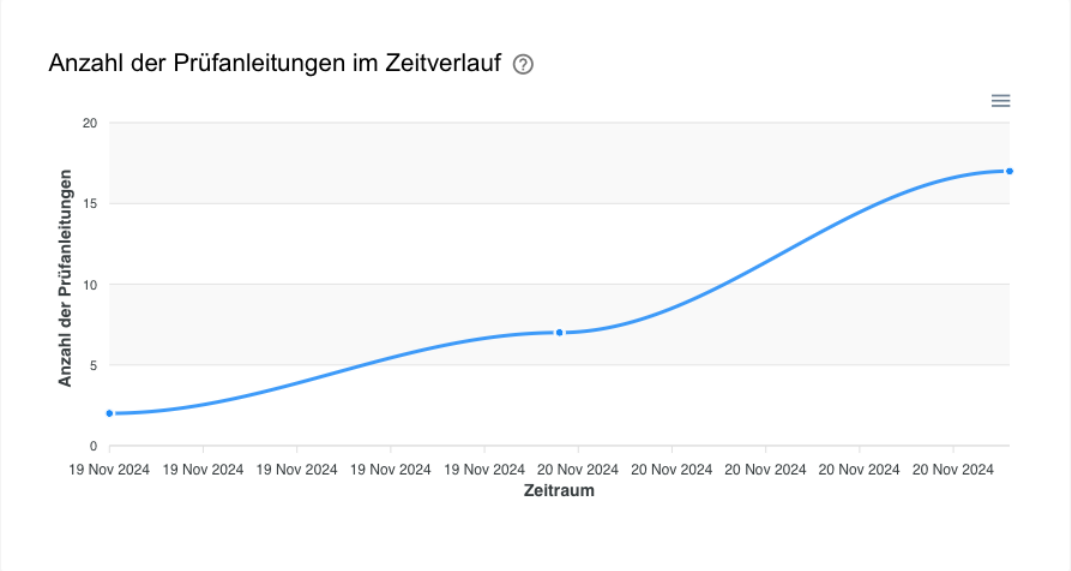


Figure 5.14.: Amount of examination guidelines over time

5.10.4. Examination Guidelines per Topic

The Examination Guidelines per Topic graph (Figure 5.15) is constructed using a horizontal bar chart. This chart is sorted by the number of examination guidelines associated with each topic, helping lawyers identify potential imbalances in rule coverage across different topics. By visualizing which topics have more or fewer examination guidelines, users can assess whether certain areas require more attention or refinement.

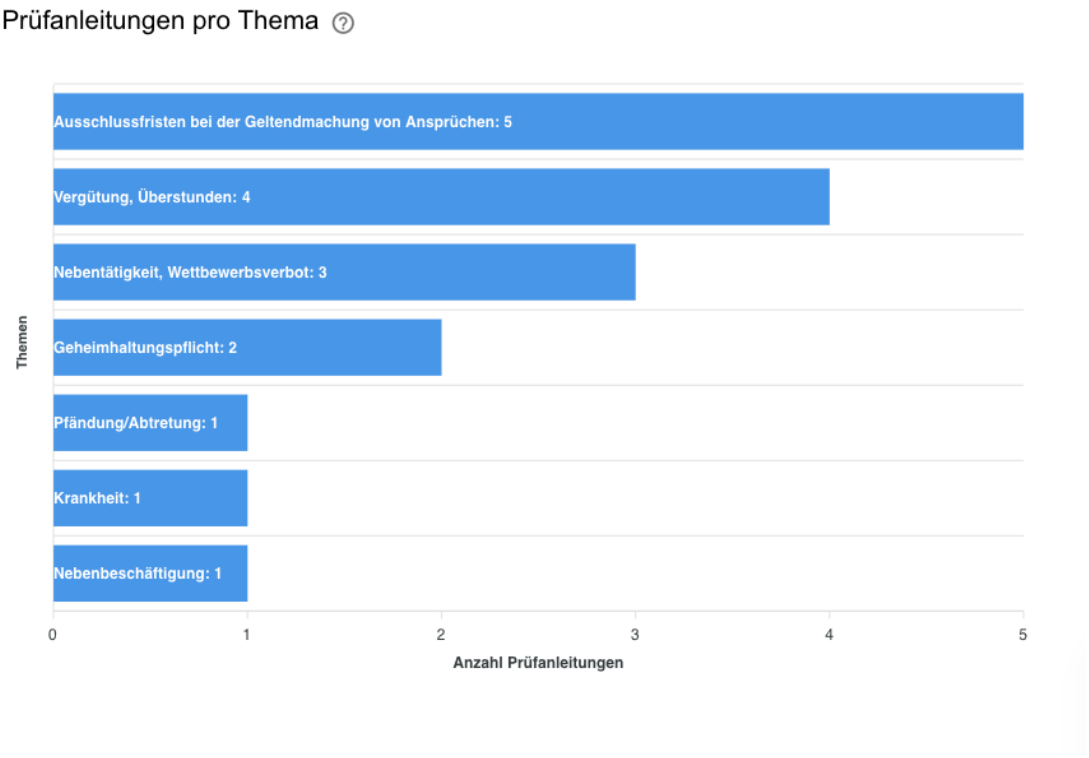


Figure 5.15.: Examination guidelines per topic

5.11. Detail Information

The Detail Information tab allows users to dive deeper into the system's evolution and performance metrics over time. This section includes several key metrics that provide more granular insights into how well the system is working.

5.11.1. LLM Accuracy over Time

The first metric in this section is the LLM Accuracy over Time (Figure 5.16), which tracks the changing accuracy of the LLM as contracts are reviewed. This metric compares the LLM evaluations with those conducted by lawyers, with the latter serving as the ground truth.

Accuracy is calculated by dividing the number of correctly evaluated clauses by the total number of LLM reviews for the relevant contracts. In cases where no human review exists for comparison, the machine review is assumed to be correct. This rolling count allows users to observe trends in accuracy over time as the system evolves.

Additionally, a filtering option has been introduced, enabling users to view the accuracy of specific examination guidelines. This feature provides granular insights into the performance of individual rules when applied by the system, which can be seen in Figure B.7.

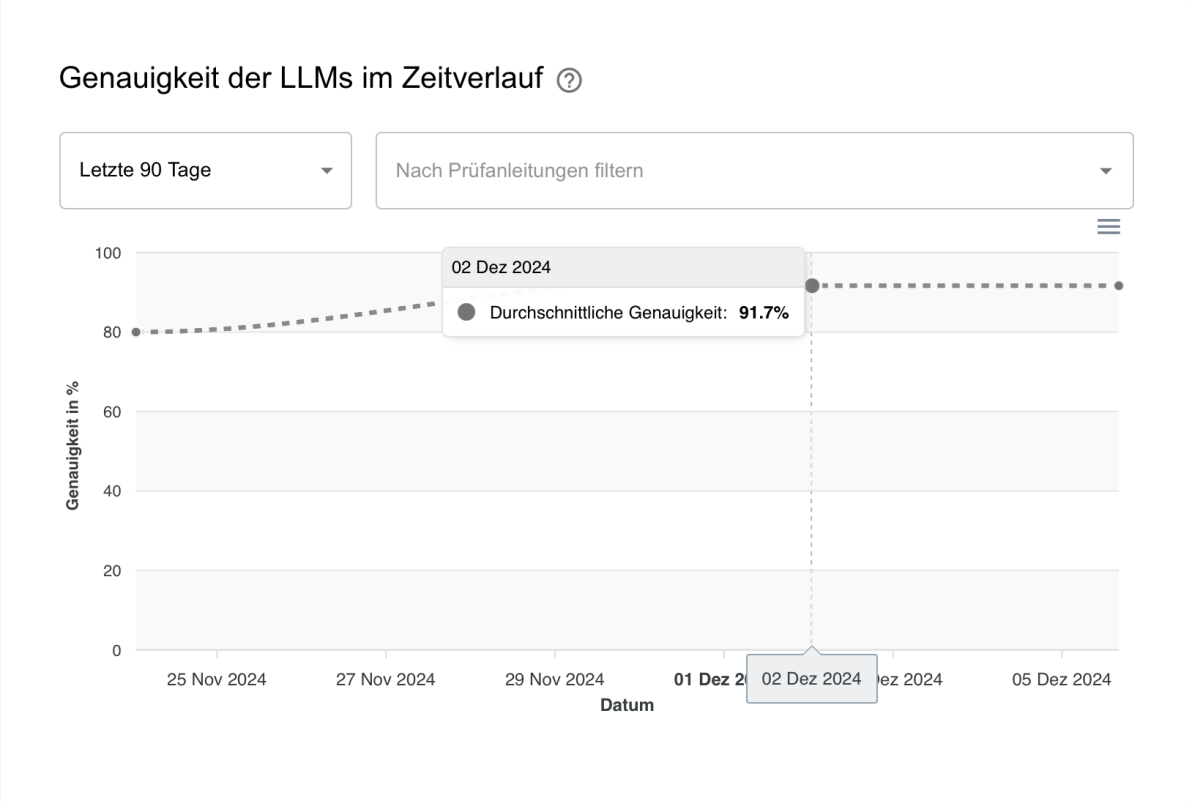


Figure 5.16.: Accuracy of the LLMs over time

5.11.2. Confidence Score

The Confidence Score metric evaluates the confidence of the LLM when applying specific examination guidelines. This metric allows for comparisons across different rules and their respective versions, offering insights into how certain the model is when applying a given rule.

The visualization uses color coding to help users quickly identify which rules have higher or lower confidence scores. Darker colors indicate higher confidence, while lighter shades signify lower confidence.

As shown in Figure 5.17, each row represents a version of a rule, while each column corresponds to a specific rule. Hovering over a rectangle displays detailed information,

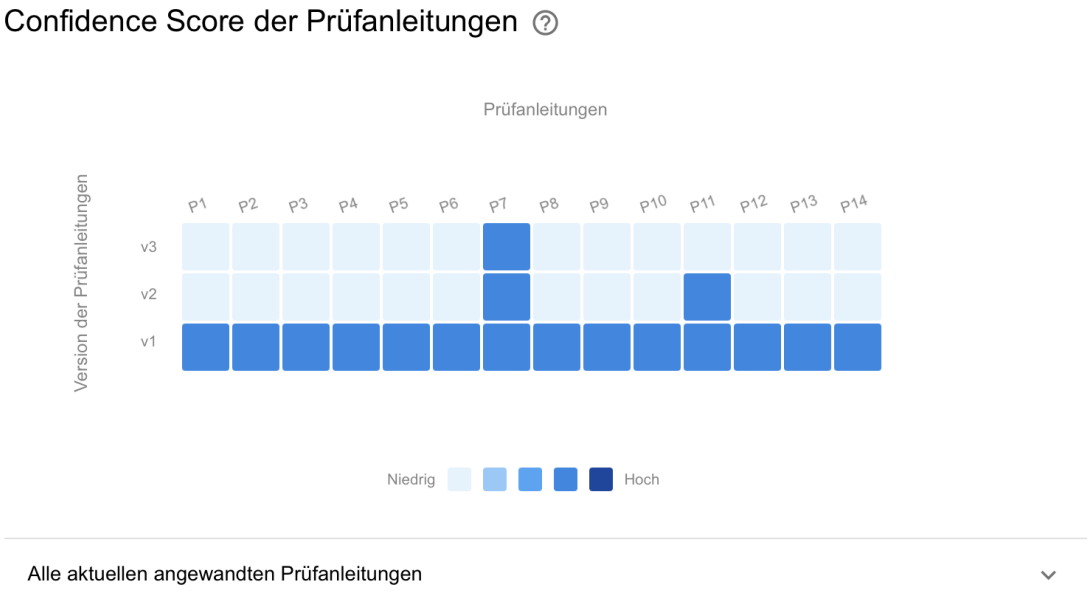


Figure 5.17.: Confidence score of examination guidelines

including the confidence score and the associated text for the version. This interactive feature enhances the user's ability to analyze and compare confidence levels across rules and versions effectively. If an examination guideline is applied regularly and the score is high, it can indicate that the examination guideline is well-written, unambiguous, and clear. On the other hand, when the confidence score for an examination guideline is low, it may signal that the rule needs optimization and should be rewritten.

5.11.3. Confusion Matrix

The final metric in this tab is the Confusion Matrix, which provides a more detailed evaluation of the system's performance compared to the accuracy score. Unlike the accuracy metric, which only distinguishes between correct and incorrect classifications, the confusion matrix offers a granular breakdown of the LLM's evaluations. This allows users to analyze in detail how accurately the system classified clauses as it can be seen in Figure 5.18).

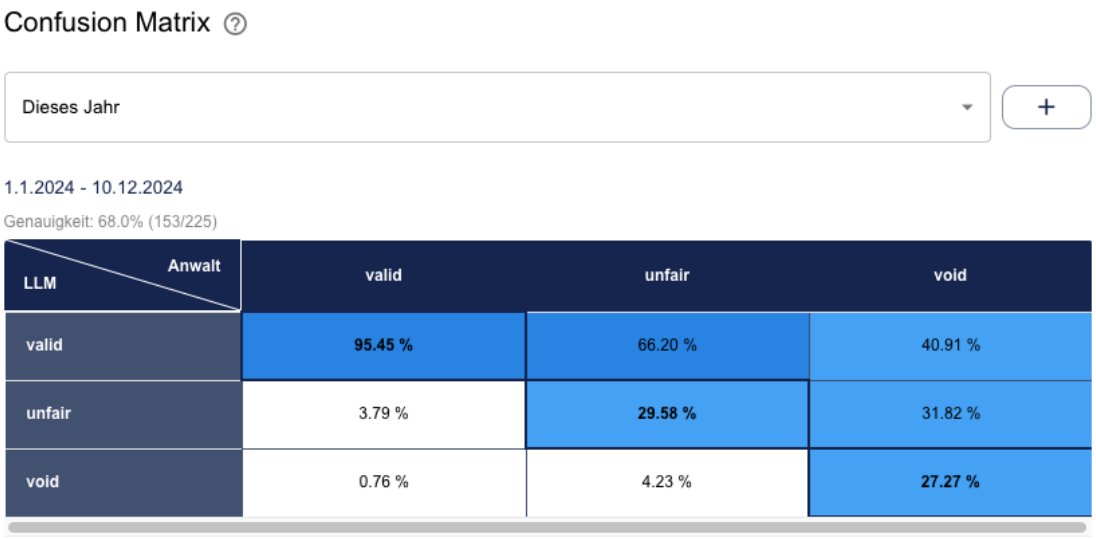


Figure 5.18.: Confusion Matrix

To enhance its usability, a comparison tool has been introduced, allowing users to evaluate system performance across different time intervals, such as:

- total
- this year
- last 90 days
- last 30 days
- last 7 days

These metrics form the foundation for evaluating and optimizing the KIBeKodA system. Offering insights into the performance of the system, which helps the lawyers to maintain and optimize the platform.

6. Summary and Outlook

This chapter provides a summary of the thesis, reflects on the lessons learned, and discusses potential improvements and future directions for the system.

6.1. Summary

The primary objectives of this bachelor thesis were achieved. The system supports editing examination guidelines and can distinguish between semantic and syntactic changes. Semantic modifications to rules are correctly recognized, triggering the system to identify clauses requiring potential reevaluation. This is possible by adjustments made to the prompt structure, resulting in a more consistent and structured LLM output.

As demonstrated in Chapter 5, the system achieves its intended functionality and resolves the lack of transparency identified at the start of the project. Technically, the system now provides valuable insights through a user-friendly analytics page. The updated system structure and architecture, excluding the analytics module, is depicted in Figure 6.1. For an overview of the complete system, including analytics, refer to Figure A.1.

The metrics derived from the literature review further contributed to a functional and intuitive interface.

While the metrics look promising, it remains to be seen whether they will have a decisive long-term impact on lawyers' workflows. Nonetheless, the prototype performs as expected and provides a solid foundation for further refinement and testing.

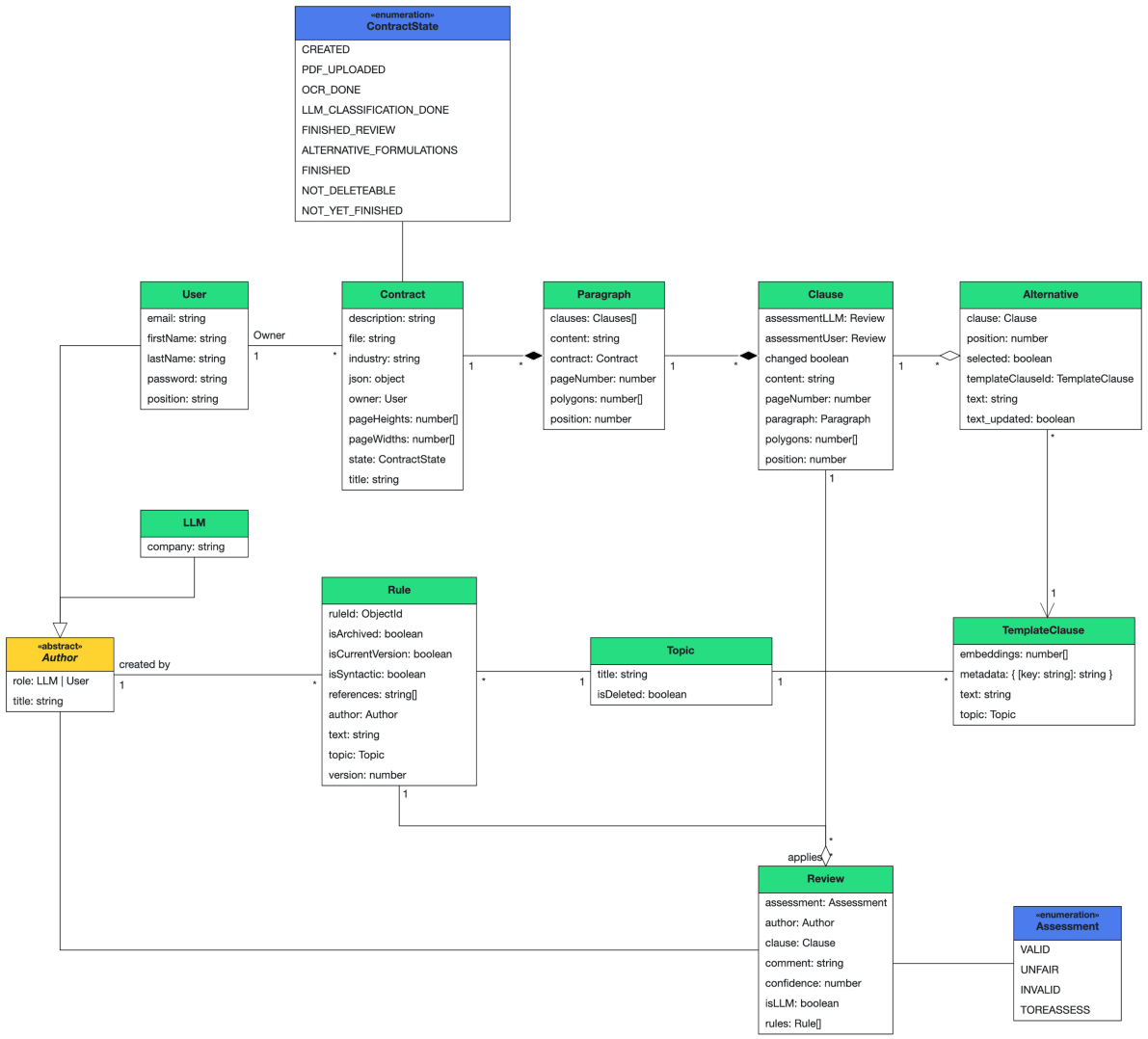


Figure 6.1.: New components of KIBeKodA without Analytics

6.2. Limitations

Various limitations contributed to the outcome of this thesis. One key factor was the tight schedule, due to the nature of a bachelor thesis, which restricted the evaluation process. Additionally, we were not able to conduct a round of feedback from the lawyers. Despite the fact that these metrics can be refined in future research, user feedback rounds during the thesis would have supported our conclusions.

The limited timeframe also impacted the ability to thoroughly test the system with actual changes in legal texts. Instead, simulated scenarios were prepared to test the system in the future. But, still a lot of internal testing gives a strong indication that the system is working

as intended. However, we still need to test KIBeKodA with real-world data to ensure the system is robust and usable in the long run. As shown in Figure B.5, certain inaccuracies remain due to LLM responses.

Adjustments to the prompt structure also introduced some variability in clause evaluations when compared to the previous system. In many cases, the system's assessments were overly lenient, as it tended to classify clauses as "valid" more often than it should. Further calibration of the prompt are needed in the future.

6.3. Outlook

The work presented in this thesis establishes a solid basis for future enhancements. Several potential directions for improvement have been identified:

First, additional metrics could be developed based on feedback from the lawyers. For example, metrics highlighting how frequently specific examination guidelines are applied to particular topics or paragraphs could help identify irrelevant or redundant rules, further optimizing the system.

To achieve this, linking topics to the paragraphs elaborated in the OCR process would be highly beneficial. Currently, paragraph titles are not semantically matched to existing topics. Mapping them could improve the pre-filtering of relevant rules based on paragraph topics, improving classification accuracy and reducing mistakes in the results. This approach would also open the door for new metrics, such as tracking rule application frequency by topic.

Additionally, displaying the LLM version used in the accuracy score visually would allow users to compare model performance more effectively and decide to switch to another version. Also, adding the option to directly switch to old LLM versions within the interface, would advocate the prior suggestion.

The confidence score metric, while conceptually useful, requires refinement. Replacing it with an accuracy-based comparison across versions could significantly enhance its usability.

Finally, conducting usability studies with a group of legal professionals would help refine both the system design and the implemented metrics. Their input can help make the system better suited to what they need. To support this process, a detailed questionnaire has been prepared within the scope of this thesis, offering a foundation for collecting structured feedback in future evaluations.

A. General Addenda

A.1. New System Architecture and Design

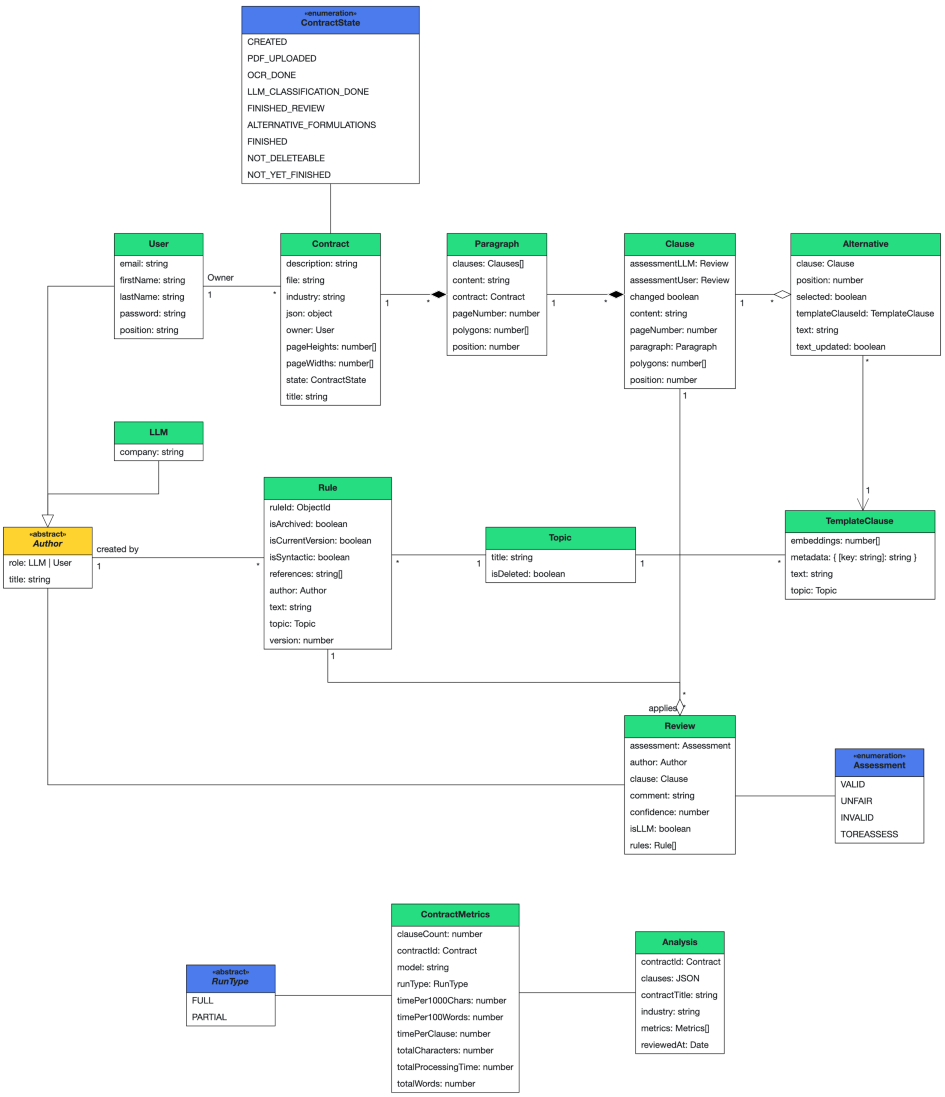


Figure A.1.: New system architecture and design

A.2. Evaluation Questionnaire

Tables A.1 and A.2 display the questions included in the evaluation questionnaire. The first column represents the question number, and the second column contains the corresponding question text. Table A.1 presents the original questions in German, while Table A.2 provides their English translations.

A.2.1. Evaluation Questions for KIBeKodA System (German)

ID	Frage
1	Haben Sie bei der Vertragsanalyse das Feld "AI Assessment" gesehen? Wenn ja, wie ist Ihr Eindruck zu zuvor?
2	Ich fand die Vertragsanalyse unnötig komplex. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
3	Ich dachte, die Vertragsanalyse war einfach zu bedienen. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
4	Ich habe mich sehr selbstsicher gefühlt, die Vertragsanalyse zu verwenden. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
5	Haben Sie eine Veränderung im Vertrag wahrgenommen? Wenn ja, welche?
6	Hat sich das System so verhalten, wie Sie es erwartet haben?
7	Wie zufrieden sind Sie mit dem neuen Feature? (1 = wenig zufrieden, 5 = sehr zufrieden)
8	Ich fand den Prozess, eine Prüfanleitung zu editieren unnötig komplex. (1 = Stimme ich überhaupt nicht zu, 5 = Stimme voll zu)
9	Ich fand diesen Teil sehr umständlich zu benutzen. (1 = Stimme ich überhaupt nicht zu, 5 = Stimme voll zu)
10	Ich fand die Analytics Seite unnötig komplex. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
11	Ich fand die Analytics Seite hilfreich. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
12	Ich fand die verschiedenen Metriken waren gut in der Analytics Seite integriert. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
13	Ich würde mir vorstellen, dass die meisten Leute sehr schnell lernen würden, die Metriken zu benutzen. (1 = stimme überhaupt nicht zu, 5 = stimme voll zu)
14	Gibt es etwas, was Sie an der Analytics Seite ändern würden? (Aufbau, Design, Metriken) Wenn ja, was?
15	Welche der Metriken halten Sie für am relevantesten?
16	Konnten Sie Ihre Aufgaben erfolgreich erledigen? Wenn nein, warum nicht?
17	Welche positiven oder auch negativen Erfahrungen haben Sie insgesamt mit der Web Applikation gemacht?
18	Was ist Ihnen besonders aufgefallen im Vergleich zu der alten Version?
19	Gibt es weitere Features, die Sie sich in Zukunft wünschen würden?
20	Gab es in der Vertragsanalyse ein Feature, welches Ihren Workflow positiv beeinflusst hat? Wenn ja, welches?
21	Gibt es Fehler im System, die Ihnen aufgefallen sind? Wenn ja, wo?

Table A.1.: Fragen zur Evaluation des KIBeKodA-Systems

A.2.2. Evaluation questions for KIBeKodA system (English)

ID	Question
1	Did you notice the "AI Assessment" field during the contract analysis? If so, what was your impression compared to before?
2	I found the contract analysis unnecessarily complex. (1 = strongly disagree, 5 = strongly agree)
3	I thought the contract analysis was easy to use. (1 = strongly disagree, 5 = strongly agree)
4	I felt very confident using the contract analysis. (1 = strongly disagree, 5 = strongly agree)
5	Did you notice any changes in the contract? If yes, what were they?
6	Did the system behave as you expected it to?
7	How satisfied are you with this new feature? (1 = not satisfied at all, 5 = very satisfied)
8	I found the process of editing a Prüfanleitung unnecessarily complex. (1 = strongly disagree, 5 = strongly agree)
9	I found this part very cumbersome to use. (1 = strongly disagree, 5 = strongly agree)
10	I found the Analytics page unnecessarily complex. (1 = strongly disagree, 5 = strongly agree)
11	I found the Analytics page helpful. (1 = strongly disagree, 5 = strongly agree)
12	I found the various metrics were well integrated into the Analytics page. (1 = strongly disagree, 5 = strongly agree)
13	I would imagine that most people would learn to use these metrics very quickly. (1 = strongly disagree, 5 = strongly agree)
14	Is there anything you would change about the Analytics page (structure, design, metrics)? If so, what?
15	Which of the metrics do you consider most relevant?
16	Were you able to complete your tasks successfully? If not, why not?
17	What positive or negative experiences have you had with the web application overall?
18	What stood out to you compared to the previous version?
19	Are there any additional features you would like to see in the future?
20	Was there a feature in contract analysis that positively impacted your workflow? If so, which one?
21	Did you notice any errors in the system? If so, where?

Table A.2.: Questions for KIBeKodA System Evaluation

B. Figures

B.1. Frontend Enhancements

This section showcases the frontend adjustments made to the system, providing a comprehensive overview of its current design and functionality.

B.1.1. Template Clause Overview with Integrated Search Bar

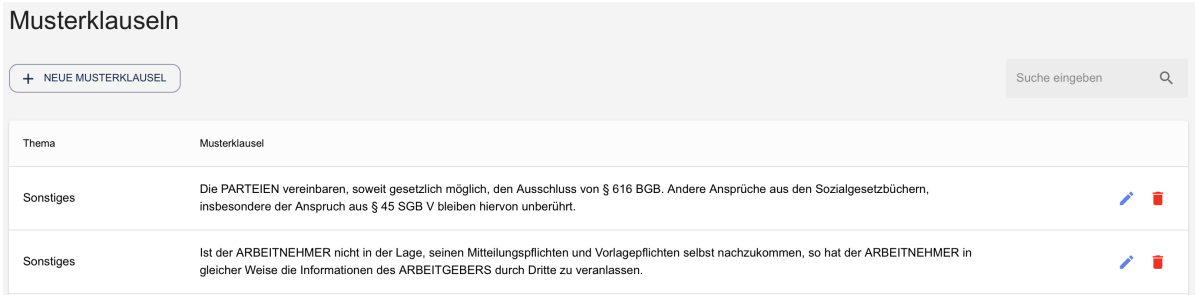


Figure B.1.: Template Clause overview with search bar

B.1.2. Version History of Examination Guidelines on the Rules Page

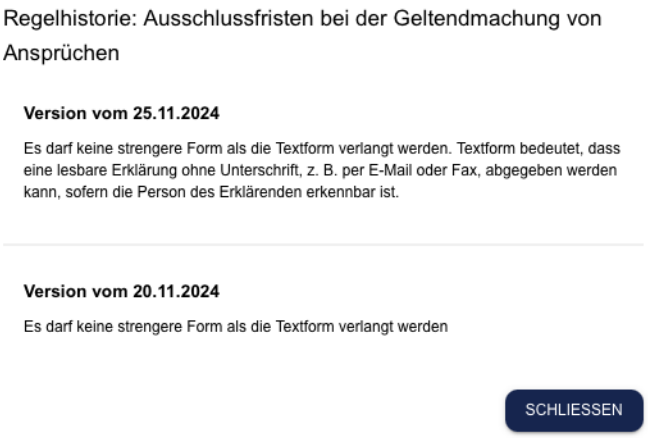


Figure B.2.: Versions of examination guideline

B.2. Contract Analysis Changes

This section spotlights specific changes made in the contract analysis, showcasing two examples of applied rules. Including a visible error to illustrate potential system limitations, such as incorrectly applied rules or overly lenient assessments.

B.2.1. Contract Analysis Header with Enhanced Filters and Reset Feature

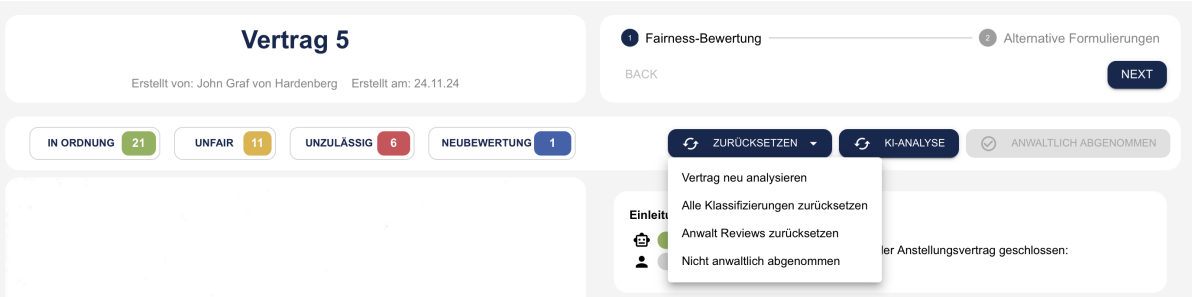


Figure B.3.: Contract analysis header with enhanced filters and reset feature

B.2.2. Contract Evaluation with Correctly Applied Rules

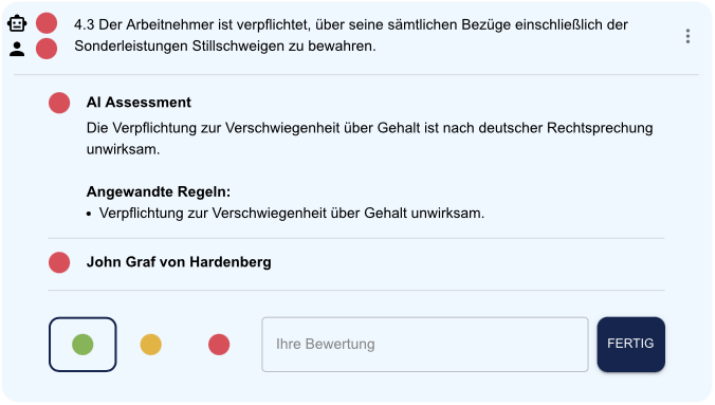


Figure B.4.: Contract evaluation with correctly applied rules

B.2.3. Contract Evaluation with Incorrectly Applied Rules

Figure B.5.: Contract evaluation with incorrectly applied rules

B.2.4. Contract Evaluation with Toreassess Clause

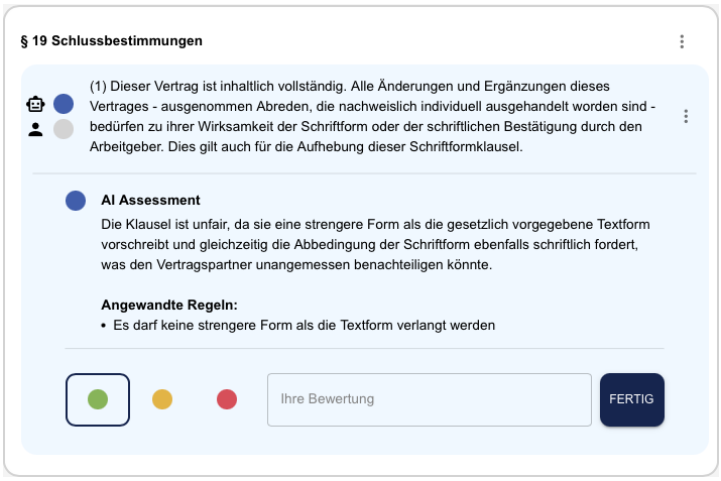


Figure B.6.: Contract evaluation with toreassess clause

B.3. Analytics Page Visualizations

This section presents additional visualizations from the analytics page, showcasing metrics and their integration into the system.

B.3.1. Accuracy Analysis for a Specific Examination Guideline

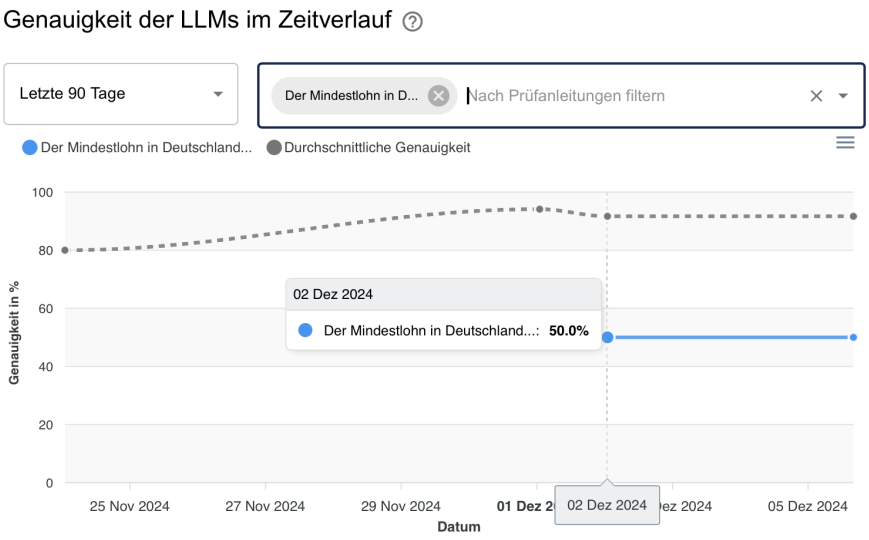


Figure B.7.: Accuracy analysis for a specific examination guideline

List of Figures

List of Tables

Glossary

ApexCharts ApexCharts is a library that allows developers to create interactive and customizable charts within React applications. 34

ChromaDB ChromaDB is an open-source vector database designed to store and manage vector embeddings for AI and machine learning applications. 27

CO-STAR CO-STAR is a structured framework for creating effective prompts in AI, developed by GovTech Singapore. It stands for Context, Objective, Style, Tone, Audience, and Response. 17

Gartner Gartner provides information about market analysis and development of AI. It is based in the USA. 2

Material UI Material UI is an open-source React component library implementing Google's Material Design, offering prebuilt components for web development. 30

MERN MERN is a technology stack in web development. It is an acronym for MongoDB, Express, React and Node.js. 4

MongoDB MongoDB is a document-oriented NoSQL database designed for modern application development with a flexible schema approach. 8, 27

MongoDB Atlas MongoDB Atlas is a fully managed cloud database service that provides a suite of tools for deploying and managing MongoDB databases in the cloud. 27

OpenAI OpenAI is an American AI research organization founded in December 2015, known for developing advanced AI models like ChatGPT. 17, 18, 23, 27

RQ1 How can changes in examination guidelines and their impact on the evaluation of contract clauses in an in-context learning AI system be tracked and assessed?. 13

Stack Overflow Stack Overflow is a forum for programmers for coding and support questions. It serves as a valuable resource for coding knowledge. 4

Acronyms

AI Artificial Intelligence. 1, 2, 3, 6, 7, 10, 11, 19, 20, 23, 26, 32, 33

API Application Programming Interface. 6

CLM Contract Lifecycle Management. 2, 10, 13

CoT Chain of Thought. 14, 24

ICL In-Context Learning. 7

JSON JavaScript Object Notation. 8

JWT JSON Web Token. 27

KIBeKodA AI-Assisted Legal Analysis and Correction of German Employment Contracts. v, vi, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21, 23, 24, 25, 26, 27, 31, 40, 42, 43, 52

LLM Large Language Model. 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 17, 18, 19, 23, 24, 25, 26, 29, 31, 32, 34, 35, 36, 37, 38, 40, 41, 43, 52, 54

NLG Natural Langauge Generation. 7

NLP Natural Language Processing. 6, 7, 8

NLU Natural Language Understanding. 7

OCR Optical Character Recognition. 7, 8, 32, 33, 43

RAG Retrieval-Augmented Generation. 14

SCD Slowly Changing Dimensions. v, vi, 14, 15, 17, 22, 23

SUS System Usability Scale. 21

UI User Interface. 6, 8, 11, 30

Bibliography

[1] Gartner. *Gartner Predicts the Global Legal Technology Market Will Reach $50 Billion by 2027 as a Result of GenAI*. Accessed: November 24, 2024. Gartner, 2024. URL: https://www.gartner.com/en/newsroom/press-releases/2024-04-25-gartner-predicts-global-legal-technology-market-will-reach-50-billion-by-2027-as-a-result-of-genai.

[2] Grand View Research. *Legal Technology Market Size, Share & Growth Report, 2030*. Accessed: November 19, 2024. 2024. URL: https://www.grandviewresearch.com/industry-analysis/legal-technology-market-report.

[3] Fortune Business Insights. *Legal Technology Market Size, Share & COVID-19 Impact Analysis, By Component (Solution and Services), By Deployment (On-premises and Cloud), By End-user (Law Firms, Corporate Legal Departments, and Others), and Regional Forecast, 2023-2030*. Accessed: November 19, 2024. 2024. URL: https://www.fortunebusinessinsights.com/legal-technology-market-109527.

[4] Gartner. *How Legal Technology Boosts Team Efficiency and Performance*. Accessed: November 19, 2024. Gartner, 2024. URL: https://www.gartner.com/en/legal-compliance/topics/legal-technology.

[5] Stack Overflow. *2024 Developer Survey*. Accessed: November 19, 2024. Stack Overflow, 2024. URL: https://survey.stackoverflow.co/2024/technology.

[6] MongoDB. *MERN Stack*. Accessed: November 19, 2024. MongoDB, 2024. URL: https://www.mongodb.com/resources/languages/mern-stack.

[7] M. S. University. *Artificial Intelligence: An Overview*. Accessed: November 19, 2024. 2021. URL: https://digitalcommons.murraystate.edu/cgi/viewcontent.cgi?article=1148%5C&context=bis437.

[8] O. E. Dictionary. *Artificial Intelligence Definition*. Accessed: November 19, 2024. 2021. URL: https://www.oed.com/dictionary/artificial-intelligence_n?tab=meaning_and_use.

[9] E. Britannica. *History of Artificial Intelligence*. Accessed: November 19, 2024. 2021. URL: https://www.britannica.com/science/history-of-artificial-intelligence.

[10] P. Journal. "Artificial General Intelligence: A Survey". In: (2020). Accessed: November 19, 2024. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7605294/.

[11] V. Archive. *Artificial Superintelligence: The Next Frontier?* Accessed: November 19, 2024. 2017. URL: https://vixra.org/pdf/1706.0468v1.pdf.

[12] IBM. *Large Language Models*. Accessed: November 19, 2024. 2023. URL: https://www.ibm.com/topics/large-language-models.

[13] IBM. *Natural Language Processing*. Accessed: November 19, 2024. 2023. URL: https://www.ibm.com/topics/natural-language-processing.

[14] S. Journal. "Natural Language Processing: An Overview". In: (2022). Accessed: November 19, 2024. URL: https://link.springer.com/article/10.1007/s11042-022-13428-4.

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[16] ABBYY. *What is OCR?* Accessed: November 19, 2024. 2023. URL: https://pdf.abbyy.com/learning-center/what-is-ocr/.

[17] IBM. *Optical Character Recognition*. Accessed: November 19, 2024. 2023. URL: https://www.ibm.com/think/topics/optical-character-recognition.

[18] R. AI. *Legal AI for Contracts: Transforming Contract Review and Analysis*. Accessed: 2024-12-10. 2024. URL: https://www.robinai.com.

[19] L. AI. *Libratech AI*. Accessed: 2024-12-10. URL: http://libratech.ai.

[20] Oracle. *Slowly Changing Dimensions*. Accessed: November 19, 2024. Oracle, 2024. URL: https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/10g/r2/owb/owb10gr2_gs/owb/lesson3/slowlychangingdimensions.htm.

[21] DataCamp. *Mastering Slowly Changing Dimensions*. Accessed: November 19, 2024. DataCamp, 2024. URL: https://www.datacamp.com/tutorial/mastering-slowly-changing-dimensions-scd.

[22] S. Teo. *How I Won Singapore's GPT-4 Prompt Engineering Competition*. 2023. URL: https://towardsdatascience.com/how-i-won-singapores-gpt-4-prompt-engineering-competition-34c195a93d41.

[23] OpenAI. *Structured Outputs*. Accessed: November 19, 2024. OpenAI, 2024. URL: https://platform.openai.com/docs/guides/structured-outputs/examples?context=ex2.

[24] K. Tian, E. Mitchell, A. Zhou, A. Sharma, R. Rafailov, H. Yao, C. Finn, and C. D. Manning. *Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback*. 2023. arXiv: 2305.14975 [cs.CL]. URL: https://arxiv.org/abs/2305.14975.

[25] K. Shen and M. Kejriwal. *A Formalism and Approach for Improving Robustness of Large Language Models Using Risk-Adjusted Confidence Scores*. 2023. arXiv: 2310.03283 [cs.CL]. URL: https://arxiv.org/abs/2310.03283.

[26] S. Thomas. *Lead time versus Cycle Time – Untangling the confusion*. 2015. URL: https://itsadeliverything.com/lead-time-versus-cycle-time-untangling-the-confusion.

[27] S. Visa, B. Ramsay, A. Ralescu, and E. Knaap. *Confusion Matrix-based Feature Selection.* Jan. 2011.

[28] P. Khayyatkhoshnevis, S. Tillberg, E. Latimer, T. Aubry, A. Fisher, and V. Mago. *Comparison of Moderated and Unmoderated Remote Usability Sessions for Web-Based Simulation Software: A Randomized Controlled Trial*. Ed. by M. Kurosu. Cham, 2022.

[29] M. Ward, A. Meade, C. Allred, G. Pappalardo, and J. Stoughton. "Careless response and attrition as sources of bias in online survey assessments of personality traits and performance". English. In: *Computers in Human Behavior* 76 (Nov. 2017), pp. 417–430. ISSN: 0747-5632. DOI: 10.1016/j.chb.2017.06.032.

[30] H.-M. Huang. "Do Print and Web Surveys Provide the Same Results?" In: *Computers in Human Behavior* 22 (May 2006), pp. 334–350. DOI: 10.1016/j.chb.2004.09.012.

[31] J. Lewis. "The System Usability Scale: Past, Present, and Future". In: *International Journal of Human-Computer Interaction* (Mar. 2018), pp. 1–14. DOI: 10.1080/10447318.2018.1455307.

[32] K. G. Guzman. *Introduction to Structured Outputs*. Accessed: November 19, 2024. 2024. URL: https://cookbook.openai.com/examples/structured_outputs_intro.

[33] G. Cloud. *Five Best Practices for Prompt Engineering*. 2023. URL: https://cloud.google.com/blog/products/application-development/five-best-practices-for-prompt-engineering?hl=en.

[34] W. P. Library. *Prompting for Complex Reasoning Tasks*. 2023. URL: https://library.westpoint.edu/GenAI/prompting.

[35] D. Docs. *Generation in RAGStack Architecture*. 2023. URL: https://docs.datastax.com/en/ragstack/default-architecture/generation.html.

[36] LoginRadius. *JWT Authentication — Best Practices and When to Use*. 2023. URL: https://www.loginradius.com/blog/engineering/guest-post/jwt-authentication-best-practices-and-when-to-use/.

[37] Snyk. *Top 3 Security Best Practices for Handling JWTs*. 2023. URL: https://snyk.io/blog/top-3-security-best-practices-for-handling-jwts/.

[38] M. Azure. *What is Middleware?* 2023. URL: https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-middleware/.

[39] MongoDB. URL: https://www.mongodb.com/products/platform/atlas-vector-search.

[40] ApexCharts. *ApexCharts - Modern Interactive Open-source Charts*. 2018. URL: https://apexcharts.com/.