Legal Knowledge and Information Systems
J. Savelka et al. (Eds.)

doi:10.3233/FAIA241230

Fighting the Knowledge Representation Bottleneck with Large Language Models

Marco BILLI [a,1,2], Giuseppe PISANO [a], Marco SANCHI [a,b,3]
[a] *CIRSFID Alma-AI, Faculty of Law, University of Bologna, Italy*
[b] *University of Pisa, Italy*

Abstract. This paper implements Large Language Models (LLMs) to support the development of expert systems in the legal domain. Our goal is to tackle one of the most critical issues related to the creation and management of rule-based systems, being the knowledge representation bottleneck. To do so, we employ GPT-4o in combination with an existing expert system developed using the Prolog language, presenting a case study based on multiple tasks. The first task deals with the formalization of legal articles in Prolog given a stable knowledge base and factual structure, including the revision of existing facts. The second task deals with the implementation of case law for updating of the expert system. To do so, it identifies the influence of case law on the application of existing norms, creates new rules and implements them in the system. This paper contributes to the field of law and Artificial Intelligence (AI) by investigating the relationship between LLMs and legal expert systems, and exploring its usefulness for knowledge engineers, as well as contributing to the research of hybrid architectures combining generative and symbolic AI.

Keywords. Large Language Models, Expert Systems, Hybrid Approach, Knowledge Representation, Prolog

1. Introduction

Expert systems have been a cornerstone of artificial intelligence (AI) for the law since their first appearance, offering structured and rule-based approaches for the automation of legal tasks. Despite their advantages in providing clear, interpretable reasoning processes, these systems face significant challenges, particularly concerning the representation, update, and scalability of their knowledge base. One of the most persistent issues is the *Knowledge Representation Bottleneck* (KRB) [1,2], which makes the acquisition, formalization, and constant update of expert knowledge a time-consuming and error-prone process. This bottleneck severely limits the flexibility and longevity of expert systems, especially when compared to state-of-the-art AI models built on vast amounts of data using machine-learning approaches.

[1]This research was funded by the European Research Council (ERC) Project "CompuLaw" (Grant Agreement No 833647) under the European Union's Horizon 2020 research and innovation program and by the European Union's Justice Programme (2022), Grant Agreement Project 101089634.
[2]All authors contributed equally to this work.
[3]Corresponding Author: Marco Sanchi, marco.sanchi4@unibo.it.

Recent advancements in *Large Language Models* (LLMs), such as GPT and its successors, offer a potential solution to overcome these limitations. In fact, LLMs show remarkable abilities in natural language understanding, contextual reasoning, and knowledge synthesis, which have opened new avenues for hybrid systems combining the interpretability of expert systems in the legal domain with the scalability and adaptability of LLMs [3,4,5].

This paper explores whether LLMs can address the next step in expert system development, particularly regarding knowledge representation, update, and scalability. Specifically, it examines whether LLMs can generate new Prolog rules in a few-shot learning context through the pipeline shown in Figure 1, thereby offering a faster and more scalable method for updating expert system knowledge bases. The *first task* investigates whether LLMs can formalize new legal rules, given a set of existing Prolog rules from an established expert system. The *second task* involves providing case examples to the LLM, asking it to extract which expand the scope of the original expert system, ensuring that they remain consistent with the original logic.

By combining these two tasks, this paper aims to demonstrate how LLMs can be leveraged to overcome the KRB, potentially transforming expert systems into more scalable and easily updatable platforms. For this study, we focus on the *Facilex* expert system[4], a rule-based system used in the application of grounds of refusal in mutual recognition instruments in the European Union, as a test-bed for evaluating the integration of LLMs [6].

This paper is structured into several key sections. It begins with Section 2, where previous research relevant to the topic is discussed, providing context and background. Next, Section 3 outlines the methodology, including validation criteria that are used to evaluate the results. The following section, Section 4, describes the process of generating rules in Prolog, supported by prompts, outputs, and a discussion of the findings. Similarly, Section 5 covers the generation of cases with accompanying prompts, outputs, and a detailed discussion. Finally, the paper closes with Section 6, summarizing the key findings and implications of the study.

2. Related Works

This section introduces related works displaying latest research regarding the interplay between expert systems and generative artificial intelligence, with particular regards to Large Language Models, focusing on the deployment of the latter for the development and support of the former.

Foremost, [4] highlights the potential of LLMs for the automated extraction of structured legal representations in an effort to support the development of legal expert systems and their potential for access to justice. The paper establishes a positive correlation between LLM-generated pathways for legal decision support systems with manually created ones, streamlining the development of symbolic legal systems through the use of LLMs. Similarly, [3] underscores the capabilities of LLMs in improving the accessibility of rule-based legal systems. Through what is described as a '*Chain of Prompts*' methodology, the paper investigates the use of LLMs for the translation of explanations

[4]https://facilex-tool.eu/

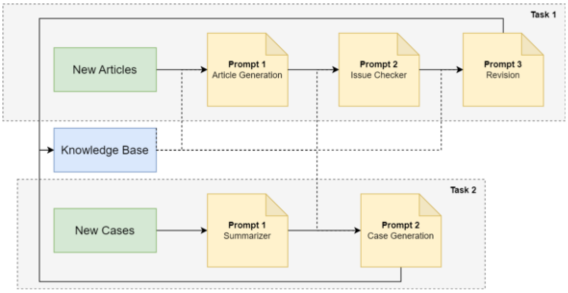

Figure 1. Pipeline for Article and Case generation.

generated by rule-based systems, from programming languages into natural language, to empower non-experts to perform complex legal tasks independently, including the comparison of different rule-based inferences. With regards to the knowledge representation bottleneck, [5,7] explore the use of Natural Language Processing (NLP) and LLMs to convert legal natural language into machine-readable rules, supporting the automated formalization of juridical knowledge, while stressing the difficulties and opportunities in converting legal documents into machine-readable rules.

Above all considered, this paper contributes to a growing field of research centered around the use of state-of-the-art natural language processing and generation for the development of expert legal systems and formalization of juridical knowledge, as well as the potential interactions of generative and symbolic AI.

3. Experimental Setup

The experiments in this study are conducted using the *Facilex Advisory Module*, a rule-based expert system designed to assist legal professionals in interpreting and applying European Union (EU) procedural directives and mutual recognition instruments. Facilex is built on a computable version of legal norms, which are translated into rules written in Prolog, a logic programming language well-suited for formal reasoning and rule-based systems. The system enables legal practitioners to define cross-border legal cases by answering a series of questions, which in turn allow the system to generate customized legal assessments. These assessments offer insights into legal remedies and potential cooperation issues relevant to the specific case at hand. The legal rules and case law used for this study include landmark EU cases and acts. Specifically, we focus on:

- Article 4 of the Council Framework Decision of 13 June 2002 on the European Arrest Warrant and the surrender procedures between Member States;
- Judgment of the Court (Grand Chamber), 16 November 2010, Case C-261/09;
- Judgment of the Court (Grand Chamber), 17 July 2008, Case C-66/08.

For the experiments, we utilize GPT-4o [8], version *GPT-4o-2024-08-06*[5], chosen due to its state-of-the-art performance across various natural language understanding and reasoning benchmarks [9]. We rely on GPT's few-shot learning capabilities to generate contextually relevant and accurate outputs from prompts, making it ideal for generating Prolog rules based on few expert examples.

[5]The latest stable model from `https://platform.openai.com/docs/models/gpt-4o`

3.1. Validation Criteria

The validation of the generated Prolog rules follows a two-tiered evaluation process: *Formal* and *Expert*. These criteria are adapted from state-of-the-art research on LLM performance evaluation[10,11], following a human in-the-loop approach for the assessment of legal tasks[4,9,12].

Formal validation of Prolog rules focuses purely on the syntactic and executable correctness of the generated rules. Specifically, this step checks whether the Prolog code produced by the model can run without errors. This process is entirely automated, ensuring that all generated rules are syntactically valid and executable within the Facilex system. However, this step does not verify the accuracy or appropriateness of the rules in a legal context.

Expert validation is carried out by the knowledge engineers and is based on the following criteria:

- **Accuracy**: The degree to which the generated Prolog rules capture all the main elements from the input legal text, ensuring that the rules are legally sound and complete, based on the expert's original formalization.
- **Relevance**: The generated rules must meet the reasoning of legal experts and adhere to the intended legal text.
- **Human Alignment**: This criterion ensures that the system supports a continuous dialogue between the model and the knowledge engineer, rather than fully automating the creation of expert systems.
- **Fluency**: The Prolog rules should exhibit a consistent and coherent style, contributing to their readability and usability by legal professionals as intended in the original expert system.

These criteria ensure that the generated rules are not only correct in form but also relevant and accurate within the specific expert system being addressed.

4. Article Generation

The first task follows the *Chain of Prompts* methodology outlined in Section 2 to assess whether GPT-4o can generate new Prolog rules in a legal expert system using few-shot learning, starting from existing rules and facts. A key distinction must be made between a fact and a legal condition: a fact represents a specific piece of information that can be input into the system, while a legal condition is a requirement derived from a positive legal rule. A condition may depend on multiple facts. The primary objective is to expand the list of rules in a Prolog knowledge base without introducing new facts. In this setup, the knowledge engineer maintains control over the system's domain by being responsible for the introduction of facts. The model's role is confined to rule generation based on provided examples, ensuring consistency and legal correctness.

4.1. Prompts

This task is formalized through a sequence of three prompts[6] (Figure 1):

[6]The Jupyter Notebook with prompts, inputs, and outputs of experiments from the paper can be found at `https://github.com/LegalMachineLab/JURIX24-fighting_krb`.

- **Prompt 1**: Given a set of existing Prolog rules and facts, along with the natural language representation of an article, the model is asked to generate new rules that formalize the article. The restriction is that no new facts may be introduced.
- **Prompt 2**: The model identifies potential mistakes in the rules by checking for common issues such as inconsistent naming, lack of correspondence between the legal text and the rules, or mixed legal conditions.
- **Prompt 3**: After the knowledge engineer selects the relevant problems identified by the model, a new prompt is issued to revise the Prolog rules and address the identified mistakes.

This approach ensures that the human is always kept in the loop, maintaining control over the legal interpretation. Although the Prolog rules generated by the model are syntactically correct, we focus on making sure that the legal reasoning behind the rules is consistent with the expectations of the domain expert.

4.2. *Output*

For the purpose of this paper, we only focus on Article 4 from the Council Framework Decision 2002/584/JHA (European Arrest Warrant). The goal was to generate the rule for *Article 4(6)*, given comma 1 of the same article as shown in Listing 1.

Listing 1: Article 4(1) of the 2002/54 FD

```
% Article 4
% Grounds for optional non-execution of the European arrest warrant
% The executing judicial authority may refuse to execute the European arrest warrant:
% 1. if, in one of the cases referred to in Article 2(4), the act on which the European
     arrest warrant is based does not constitute an offence under the law of the executing
     Member State; [...];

optional_refusal(article4_1, ExecutingMemberState, europeanArrestWarrant):-
    eaw_matter(PersonId, IssuingMemberState, ExecutingMemberState, Offence),
    art2_4applies(Offence),
    national_law_not_offence(Offence, ExecutingMemberState).

eaw_matter(PersonId, IssuingMemberState, ExecutingMemberState, Offence) :-
    ...
```

Listing 2 shows the rules generated by the model using **Prompt 1**. They deal with the refusal of an arrest warrant if the requested person is a resident, national, or staying in the executing Member State and if that state undertakes to execute the custodial sentence or detention order.

Listing 2: Article 4(6) of the 2002/54 FD - Generated by GPT-4o

```
%6. if the European arrest warrant has been issued for the purposes of execution of a
     custodial sentence or detention order,
eaw_for_execution(PersonId, ExecutingMemberState):-
    executing_proceeding(ExecutingMemberState, PersonId, execution_custodial_sentence)
    ; executing_proceeding(ExecutingMemberState, PersonId, execution_detention_order).

%where the requested person is staying in, or is a national or a resident of the executing
     Member State
person_in_executing_state(PersonId, ExecutingMemberState):-
    person_staying_in(PersonId, ExecutingMemberState)
    ; person_nationality(PersonId, ExecutingMemberState)
    ; person_residence(PersonId, ExecutingMemberState).
```

```
12 %and that State undertakes to execute the sentence or detention order in accordance with its domestic law;
13 state_undertakes_execution(ExecutingMemberState, Offence):-
14     executing_proceeding_status(Offence, ExecutingMemberState, execution_undertaken).
15
16 %optional refusal based on article 4(6)
17 optional_refusal(article4_6, ExecutingMemberState, europeanArrestWarrant):-
18     eaw_matter(PersonId, IssuingMemberState, ExecutingMemberState, Offence),
19     eaw_for_execution(PersonId, ExecutingMemberState),
20     person_in_executing_state(PersonId, ExecutingMemberState),
21     state_undertakes_execution(ExecutingMemberState, Offence).
```

Upon evaluation, it was found that the expert's original representation of *Article 4(6)* did not include the 'state_undertakes_execution' condition, as this is considered implicit in the domain. The condition is slightly redundant if one considers the conditions present in Listing 2 lines 3 and 4. The condition 'eaw_matter' has been created by the knowledge engineer to state that a proceeding is ongoing between an issuing and executing state, and one of the conditions for the grounding of this predicate consists in verifying that a proceeding has started in both states. Furthermore, GPT modelled the subrule 'eaw_for_execution', while the knowledge engineer kept the two subconditions in the code. However, they are the same, therefore the rule is correctly applied.

Starting from this output, we apply Prompt 2 to obtain a list of possible issues, as shown in Listing 3:

Listing 3: Ouput Prompt 2 - Generated by GPT-4o

```
problem 1
Inconsistent naming: [...]
problem 2
Inconsistent naming: [...]
problem 3
Fail to capture relevant information: [...]
problem 4
No correspondence between natural language and rules: [...]
problem 5
Mixed legal conditions: The rule ‘state_undertakes_execution/2‘ uses the predicate ‘executing_proceeding_status/3‘, which may not clearly represent the legal condition of a state undertaking to execute a sentence according to its domestic law. This could lead to incorrect application of legal conditions.
```

From this the user can select the relevant issues. In this case we will only consider problem 5. The model will then revise and apply the revisions directly in the text, as shown in Listing 4.

Listing 4: Output Prompt 3 - Generated by GPT-4o

```
% Revised predicate to better represent the legal condition of a state undertaking to execute a sentence according to its domestic law
state_undertakes_execution(ExecutingMemberState, Offence):-
    domestic_law_execution_commitment(ExecutingMemberState, Offence).

% New predicate to represent the commitment of the executing state to undertake execution according to its domestic law
domestic_law_execution_commitment(ExecutingMemberState, Offence):-
    executing_member_state(ExecutingMemberState),
    offence_type(Offence),
    member(Offence, [offence1, offence2, offence3]). % Replace with actual offences that the state commits to execute
```

In general, we observed a tendency to identify as problematic those predicates that did not fully align with the terms used in the natural language rule. Additionally, we noted the creation of multiple sub-rules designed to expand the meaning and introduce further conditions to already existing predicates. Finally, we found that the model does not maintain consistency in its repetition, even when operating with a low temperature. While the overall structure remains largely uniform (i.e., the conditions in the final rule remain consistent), the names of the predicates and the number and scope of the sub-rules vary.

4.3. Discussion

The results of the validation process show that the generated Prolog rules passed the formal validation step, with all rules being syntactically and grammatically correct. The facts used in the rules matched those from the expert's original system, confirming that the rules were executable and aligned structurally with the original.

In the expert validation, the results reveal some minor issues with accuracy. While most of the expert's formalized conditions were captured, there were two redundant conditions added, two instances where negative conditions were used instead of positive ones, and a case where the system generated three separate rules instead of one rule with multiple conditions in an OR structure. Despite these discrepancies, the overall accuracy remained high, as the essential legal reasoning was correctly reflected; out of the 27 conditions modelled by the LLM, the human expert equally represented 23 of those.

The relevance of the generated rules was fully achieved, as all necessary legal conditions were present, meeting the expert's expectations. Furthermore, the logical connections used to link the conditions were correct and aligned with the intended legal reasoning. The system accurately captured the relationships between conditions, ensuring that the rules reflected the appropriate logical structures, such as AND/OR connections. This maintained the integrity of the legal framework, even when the model introduced minor variations in rule structure, such as separating multiple conditions into distinct rules.

Fluency was also satisfactory, with the Prolog rules following the stylistic guidelines provided by the expert. The names of the predicates and the structure of the arguments were consistent with the examples given, and the system demonstrated the ability to explain newly generated elements clearly. However, potential issues were identified in the context of following the style of the knowledge engineer: 1) the model tends to generate subrules to maintain the Prolog code's syntactic and semantic alignment with the natural language source; 2) the use of negation by failure in Prolog can produce unintuitive code or may not align with the style typically used by human programmers.

Human alignment was facilitated through the iterative use of multiple prompts, enabling an interactive dialogue between the model and the knowledge engineer. The process allowed the expert to refine the output progressively, ensuring that any issues were resolved through successive revisions. An issue that arose at this stage is that multiple prompts are needed to surface all potential problems.

In conclusion, the experiments suggest that GPT-4o is capable of generating coherent and consistent Prolog rules based on few-shot examples, though continuous human supervision is essential to ensure legal correctness and proper alignment with the expert system's domain.

5. Case Generation

Following the same *Chain of Prompts* approach, the second task focuses on summarizing case law and identifying key elements relevant to the domain of the legal expert system. The goal is to extract the essential legal principles and conditions from the case law, which can then be formalized into Prolog rules. The primary objective is to create Prolog rules that accurately represent the identified elements without altering the existing structure or introducing new concepts beyond the case law, which would fall outside the scope of the expert system. In this setup, the knowledge engineer defines the style and framework for rule creation, ensuring consistency across the system. The model's role is to generate rules that adhere to this style, ensuring both legal accuracy and alignment with the system's existing logic.

5.1. Prompts

The task has been divided in two subtasks (Figure 1):

- **Prompt 1**: First, we extract a summary of the conditions related to the application of a refusal ground under European law based on case law. The output should include the case law name, the relevant ground of refusal article, the conditions for refusal, an explanation of key elements, and any case law influences.
- **Prompt 2**: Based on the previous summary, provide a formalization of these elements into new rules, only including articles that relate to the ground of refusal, ignoring elements that extend the article's scope for legislators. Finally, we verify that the previous output aligns with the current rules. The final output should contain the applied article, the new Prolog rule, and an explanation.

The Facilex system models the integration of case law conditions into existing legal rules by generating a representation of the case law and embedding it within the structure of the corresponding EU legal provisions. Specifically, the system first creates a rule that captures the conditions derived from relevant case law. This rule is then combined with an existing rule that represents the EU regulation governing the ground of refusal. By linking these rules, the system ensures that the case law conditions are applied consistently with the article on which the case is based on. This is shown in Listing 6.

An alternative representation could involve modifying the EU rule, including specific exceptions taken from the case law, where it introduces additional conditions to the EU legal provisions.

5.2. Output

We will now show in Listing 5 an excerpt of the summary, displaying the focus on the key legal elements. As input we have the raw text of the Judgement taken in Case C-66/08, which clarifies the scope of application of Article 4(6).

Listing 5: Output Prompt 1 Task 2 - Generated by GPT-4o

```
- **Ground of Refusal Article**
  - Article 4(6) of Council Framework Decision 2002/584/JHA

- [...]
```

```
5
6 - **Case Law Influences**
7   - **Overall Assessment**: The executing judicial authority must make an overall assessment
       of various objective factors characterizing the situation of the requested person,
      including the length, nature, and conditions of their presence, and the family and
      economic connections they have with the executing Member State.
8   - **Non-Interruption and Compliance**: The fact that the requested person's stay in the
      executing Member State was not uninterrupted [...] determining whether the person is
       "staying" in the executing Member State.
9   - **Criminal Activity and Detention**: [...]
```

As shown in line 2, the model successfully identifies the relevant ground of refusal. The corresponding Prolog representation is provided as input in the subsequent Prompt 2, alongside an example of how case law is integrated into the expert system. Finally, a Prolog rule that accurately captures the identified conditions is generated, as illustrated in Listing 6.

Listing 6: Output Prompt 2 Task 2 - Generated by GPT-4o

```
### Article Applied
Article 4(6) of the European Arrest Warrant Framework Decision

### New Prolog Rule
```prolog
%%% Case Law Name
case_law(article4_6, CaseLawName):-
    overall_assessment(PersonId, ExecutingMemberState),
    non_interruption_and_compliance(PersonId, ExecutingMemberState).

%%% Prolog Article
case_law_refusal(article4_6, CaseLawName, europeanArrestWarrant):-
    optional_refusal(article4_6, ExecutingMemberState, europeanArrestWarrant),
    case_law(article4_6, CaseLawName).

% overall_assessment(PersonId, ExecutingMemberState) - overall assessment of the requested
     person's situation
overall_assessment(PersonId, ExecutingMemberState):-
    person_presence(PersonId, ExecutingMemberState, Length, Nature, Conditions),
    family_connections(PersonId, ExecutingMemberState),
    economic_connections(PersonId, ExecutingMemberState).

% non_interruption_and_compliance(PersonId, ExecutingMemberState) - assessment of the
     continuity and legality of the person's stay
non_interruption_and_compliance(PersonId, ExecutingMemberState):-
[...]
```

Explanation of the Conditions
[...]
```

5.3. Discussion

The formal validation of the generated Prolog rules confirmed that all rules were syntactically and grammatically correct. Given that there were no specific constraints on the facts to use, this task was made easier for the model, allowing it more flexibility in generating executable rules without encountering structural issues.

In the expert validation, the accuracy of the rules was strong. The model successfully identified and represented all key legal elements within the Prolog formalization. Each influence from case law was properly represented by distinct predicates, and sub-rules were created to provide the full specifications for each case law influence. This ensured that the formalized rules captured the necessary legal complexity.

Achieving relevance in the generated rules required substantial prompt engineering efforts, as described below. The final output contains all required legal conditions were present, and the model correctly adhered to the legal reasoning expected by the expert. Additionally, the logical connections between conditions, such as the proper use of boolean connectors, were implemented accurately, maintaining the logical flow of the legal arguments.

However, similar to Task 1, we encountered challenges with the model's tendency to introduce redundant or irrelevant elements into the output. These elements, while legally accurate, are not suitable for representation in an expert system. For instance, concepts such as 'same acts', defined autonomously under European Union law to ensure uniform application across Member States, and 'Cooperation and Information Exchange', which pertains to judicial authorities' communication across jurisdictions, were identified as unnecessary for this specific task. These concepts, although important for legal reasoning, extend beyond the scope of the expert system and therefore we worked towards their exclusion from the summary and its Prolog representation.

Fluency was maintained throughout, as the model followed the expert's stylistic guidelines closely. The predicates were named in a manner that resembled natural legal language, and the arguments were structured according to the provided examples. Furthermore, the model effectively created distinct rules for both the case law and its integration with the original legal rule. To improve the model's performance, we also reduced the size of the input summary. When presented with excessive information, the system was prone to structural errors in the output, leading to inaccuracies in the Prolog rule generation.

In terms of human alignment, the predicates generated to represent new legal concepts were well-explained, making the output accessible to the knowledge engineer. The first prompt's output could easily be refined – or completely replaced – by the legal expert, streamlining the collaboration process.

Overall, GPT demonstrates strong proficiency in extracting key and relevant legal elements from case law. The model consistently identifies the core legal principles required for generating Prolog rules that accurately represent the ground of refusal. Furthermore, the Prolog rules generated follow the style and structure established by the knowledge engineer, ensuring consistency with the expert system's requirements.

6. Conclusions

This paper shows the use of LLMs to overcome the KRB for the development of legal expert systems. It does so through a case study comprised of two tasks: automatically formalizing legal articles, and expanding the scope of the expert system through case examples.

Key findings include the opportunity to deploy LLMs to formalize and update rule-based systems, the potential for machine-human interaction to address the challenges of knowledge representation, and the effectiveness and potential of hybrid approaches, that is, those combining the interpretability and precision of expert systems with the natural language generation capabilities of LLMs.

Together with recent research which has worked on translating code into natural language, this paper fosters the development of global systems, transitioning from *text to code and back to text*, thus aiming for dynamic, scalable and user-friendly expert systems.

References

[1] Ashley K. Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age. Cambridge University Press; 2017.

[2] Santosuosso A, Pinotti G. Bottleneck or Crossroad? Problems of Legal Sources Annotation and Some Theoretical Thoughts. Stats. 2020;3(3):376-95. Available from: https://www.mdpi.com/2571-905X/3/3/24.

[3] Billi M, Parenti A, Pisano G, Sanchi M. A hybrid approach for accessible rule-based reasoning through large language models. 18th International Workshop on Juris-Informatics. 2024;Jurisin. Preprint Version. Available from: https://doi.org/10.48550/arXiv.2311.11811.

[4] Janatian S, Westermann H, Tan J, Savelka J, Benyekhlef K. From Text to Structure: Using Large Language Models to Support the Development of Legal Expert Systems. In: Sileno G, Spanakis J, van Dijck G, editors. Legal Knowledge and Information Systems - JURIX 2023: The Thirty-sixth Annual Conference, Maastricht, The Netherlands, 18-20 December 2023. vol. 379 of Frontiers in Artificial Intelligence and Applications. IOS Press; 2023. p. 167-76. Available from: https://doi.org/10.3233/FAIA230962.

[5] Nguyen HT, Goebel R, Toni F, Stathis K, Satoh K. LawGiBa - Combining GPT, Knowledge Bases, and Logic Programming in a Legal Assistance System. In: Sileno G, Spanakis J, van Dijck G, editors. Legal Knowledge and Information Systems - JURIX 2023: The Thirty-sixth Annual Conference, Maastricht, The Netherlands, 18-20 December 2023. vol. 379 of Frontiers in Artificial Intelligence and Applications. IOS Press; 2023. p. 371-4. Available from: https://doi.org/10.3233/FAIA230991.

[6] Billi M, Parenti A. Access to Justice through AI. In: Facilitating Judicial Cooperation in the EU. A computable approach to mutual recognition in criminal matters. Brill Nijhoff; forthcoming. p. 1-15.

[7] Dragoni M, Villata S, Rizzi W, Governatori G. Combining Natural Language Processing Approaches for Rule Extraction from Legal Documents. In: Pagallo U, Palmirani M, Casanovas P, Sartor G, Villata S, editors. AI Approaches to the Complexity of Legal Systems - AICOL International Workshops 2015-2017: AICOL-VI@JURIX 2015, AICOL-VII@EKAW 2016, AICOL-VIII@JURIX 2016, AICOL-IX@ICAIL 2017, and AICOL-X@JURIX 2017, Revised Selected Papers. vol. 10791 of Lecture Notes in Computer Science. Springer; 2017. p. 287-300. Available from: https://doi.org/10.1007/978-3-030-00178-0_19.

[8] OpenAI. Hello GPT-4o; 2024. Accessed: 2024-09-12. Available from: https://openai.com/index/hello-gpt-4o/.

[9] Guha N, Nyarko J, Ho DE, Ré C, Chilton A, Aditya K, et al. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023; 2023. Available from: http://papers.nips.cc/paper_files/paper/2023/hash/89e44582fd28ddfea1ea4dcb0ebbf4b0-Abstract-Datasets_and_Benchmarks.html.

[10] Chang Y, Wang X, Wang J, Wu Y, Yang L, Zhu K, et al. A Survey on Evaluation of Large Language Models. ACM Trans Intell Syst Technol. 2024;15(3):39:1-39:45. Available from: https://doi.org/10.1145/3641289.

[11] Liang P, Bommasani R, Lee T, Tsipras D, Soylu D, Yasunaga M, et al. Holistic evaluation of language models. arXiv preprint arXiv:221109110. 2022.

[12] Lam KY, Cheng VC, Yeong ZK. Applying Large Language Models for Enhancing Contract Drafting. In: LegalAIIA@ ICAIL; 2023. p. 70-80.